Grado en Ingeniería en Tecnologías de Telecomunicación

Curso 2014-2015

TRABAJO FIN DE GRADO

Plataforma web para gestión de información de sensores en entornos de UAVs

Autor: Lucía Oliva Bállega

Director: Iván Vidal Fernández
Co-director: Francisco Valera Pintor

Leganés, Junio 2015

Título: Plataforma web para gestión de información de sensores en entornos de UAVs
Autor: Lucía Oliva Bállega
Director: Francisco Valera Pintor
Tutor: Iván Vidal Fernández

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

Ahora mismo, vosotros sois lo nuevo. No permitáis que el ruido de las opiniones ajenas silencie vuestra voz interior. Y más importante todavía, tened el valor de seguir vuestro corazón e intuición, porque de alguna manera ya sabéis lo que realmente queréis llegar a ser. Todo lo demás es secundario.

Steve Jobs (1955-2011)

Agradecimientos

Me gustaría dar las gracias a los becarios del despacho 4.1.A03 del Departamento de Telemática por las anécdotas de los lunes por la mañana y a los técnicos por mejorar mis habilidades motrices.

Gracias a mis tutores, Iván y Paco, por su exigencia, apoyo constante y paciencia.

Gracias a mi madre, por soportar mis días de estrés y mis quejas infinitas.

A mi amigos Laura y Perri, porque aunque nos separan miles de kilómetros, ambos me animan cada día desde la distancia.

A Rebe, por ser la mejor compañera de cacharrería que nunca podría haber encontrado y por los retos que nos esperan juntas.

Y a mi compañero de viaje Fer, por estar siempre a mi lado recordándome que soy capaz de superar todo obstáculo que se me ponga por delante, muchas gracias.

Resumen

En plena revolución tecnológica en la que cada día aparecen nuevos dispositivos electrónicos, nos encontramos con los drones. Estos vehículos aéreos no tripulados poco a poco van ganando mayor protagonismo gracias a los avances en electrónica y al interesante abanico de sensores de los que están dotados.

El presente proyecto tiene como objetivo el diseño e implementación de una plataforma capaz de gestionar la recogida, encapsulación y entrega de los datos provenientes de estos sensores a un equipo de destino. Además, se busca que la configuración tanto de los sensores instalados como de los flujos de datos creados, sea totalmente flexible y accesible mediante una plataforma web.

Para alcanzar este objetivo se han desarrollado dos aplicaciones distintas, una que opera en el vehículo aéreo no tripulado y se encarga de ejecutar los procesos de adquisición, encapsulación y envío de datos; y otra que da soporte a la interfaz web, la cual permite al usuario configurar las operaciones precisas relacionadas con los procesos anteriormente citados.

En este documento se van describiendo las distintas fases que ha seguido el desarrollo del proyecto, partiendo del estudio sobre los dispositivos y tecnologías asociadas al mismo, pasando por su diseño e implementación, hasta alcanzar el resultado obtenido.

Palabras clave: UAV, sensores, telemetría, acceso web.

Abstract

In the midst of a technological revolution where each day new electronic devices appear, we find the drones. These unmanned aerial vehicles are gaining prominence day by day thanks to the advances in electronics and to the interesting range of sensors that can be deployed.

The aim of this project is to design and implement a platform capable of managing the sensor data reception, encapsulation and sending to a destination host. Moreover, the support of the installed sensors and the created data streams is required to be completely flexible and accessible through a web platform.

In order to reach this goal, two different applications have been developed, one that runs at the unmanned aircraft vehicle, which is in charge of executing the acquisition, encapsulation and delivery processes of sending data; and a second one that supports a web-based interface that allows the user to configure the appropriate operation of the aforementioned processes.

The current document describes the different stages corresponding to the project development, from the theoretical study of related devices and technologies, to the platform design and implementation.

Keywords: UAV, sensors, telemetry, web access.

Información complementaria

De forma previa a la lectura del presente Trabajo Fin de Grado se quiere explicar el motivo de su escritura en dos idiomas.

Según se indica en la normativa:

"Los alumnos que estén cursando Grados del Plan 2011 (y el Grado en Tecnologías de Telecomunicación) ***deberán redactar en inglés*** *de forma obligatoria en la memoria del TFG un resumen extendido (5-10 páginas), los capítulos de introducción y conclusiones (siendo recomendable la escritura de la memoria completa en inglés) para cumplir con los objetivos de aprendizaje marcados en la ficha de la asignatura TFG"*

Por ello, tanto los capítulos de Introducción y Conclusiones como el resumen extendido se han incluido en inglés al final de la memoria en forma de anexos.

Índice general

Índice de figuras

Capítulo 1

Introducción

En el presente capítulo se explican los conceptos principales detrás del proyecto, los cuales han motivado el desarrollo del sistema. Después, se detallan los objetivos que se desea alcanzar y finalmente se introduce la estructura de la memoria.

1.1. Motivación del proyecto

Hoy en día, hablar sobre drones o UAVs (del inglés, *Unmanned Aircraft Vehicle*, vehículo aéreo no tripulado) es hablar sobre una nueva idea de monitorización de la naturaleza, seguridad y misiones militares, entre otros aspectos.

Un UAV puede tener diferentes alcances (tácticos, MALE o HALE) y capacidades, y está subordinado a una estación de control en tierra (en inglés, *Ground Control Station*, GCS). Además, cada vehículo aéreo no tripulado puede llevar distintas cargas, dependiendo de la misión que tenga que realizar el sistema completo; así pues, un UAV puede tener integrado un sensor de temperatura, cámaras de video o infrarrojas, radares de apertura sintética, sensores RFID (*Radio Frecuency Identification* o identificación por radiofrecuencia) u otro tipo de equipamiento electrónico. La información generada por cada sensor o dispositivo en los distintos UAV es recogida y procesada por una unidad de adquisición de datos que la envía como telemetría a la GCS desde donde se retransmite hacia diferentes equipos encargados de analizar dichos datos.

Una vez introducido el funcionamiento de los elementos que integran un UAV, es interesante comentar cuáles son sus aplicaciones. Dentro del campo militar, el uso de estos vehículos ha significado una revolución en cuanto a las misiones de búsqueda y rescate, gestión y prevención de desastres naturales y la posibilidad de disponer

de dispositivos de vigilancia durante las 24 horas del día en aquellas áreas de difícil acceso para un avión tripulado.

Además de la gran cantidad de beneficios que trae consigo el uso de UAVs en el mundo militar, han surgido usos civiles sorprendentes relacionados con la tecnología UAV desde que esta ha sido dada a conocer, por diferentes medios, a la opinión pública, que ha comenzado a averiguar su existencia y utilidad. Este hecho junto con la evolución tecnológica de los últimos años están haciendo posible la utilización de unidades UAV en detección de huracanes, creación de mapas en 3D, protección de la naturaleza y operaciones de búsqueda y rescate.

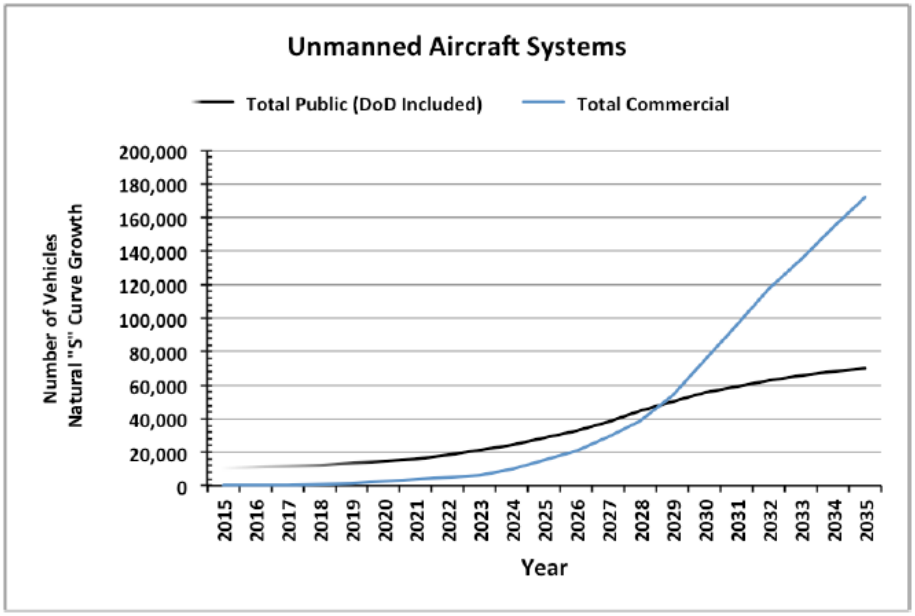

Figura 1.1: Predicción de UAS entre 2015 - 2035[1]

La gráfica de la figura 1.1 muestra un pronóstico sobre la evolución el número de UAVs en el espacio aéreo de Estados Unidos en los próximos 20 años, según un estudio realizado por el Departamento de Transporte de EEUU. Según se observa en la figura, en 2035 se espera que el número de estos dispositivos ronde los 180.000 solo en el citado país. Este dato revela cómo el campo de los UAV está todavía por explotar y supone la principal motivación del desarrollo de un proyecto como el presentado en esta memoria.

En el futuro, esos miles de UAVs producirán una enorme cantidad de información que necesitará de equipos software y hardware para

[1] Archivo. Departamento de Transporte del Gobierno de EEUU. https://fas.org/irp/program/collect/service.pdf

ser manipulada. En la actualidad, las soluciones que existen en el mercado son privadas y costosas, además de difíciles de gestionar y actualizar. Ejemplo de ello son los sistemas presentados por Ixion Industry and Aerospace [1] y la constructora naval francesa DCNS [19].

En este contexto se desarrolla el presente Trabajo Fin de Grado, que tiene por finalidad diseñar e implementar una plataforma software de bajo coste para configurar y ejecutar tareas de recogida, procesado y envío de datos procedentes de los sensores integrados en distintos UAVs, de tal forma que sea extensible y suficientemente flexible para garantizar la integración de sensores de cualquier tipo.

Finalmente, cabe destacar que Trabajo Fin de Grado surge como una línea de desarrollo futura identificada al término del proyecto DRONE (DN8644-COINCIDENTE-10032110042), financiado por el Ministerio de Defensa y realizado conjuntamente por el Instituto Nacional de Técnica Aeroespacial (INTA), Erzia Technologies y la Universidad Carlos III de Madrid.

1.2. Objetivos

En este proyecto, el objetivo principal es el diseño y desarrollo de una plataforma de recogida y envío de información de sensores de un sistema UAV. Este objetivo principal se divide en varios subobjetivos, los cuales son presentados a continuación:

- El sistema se fundamentará en protocolos y estándares actuales de Internet.
- La plataforma soportará la ejecución de procesos para la recogida, encapsulación y envío de información de los sensores de un UAV a múltiples destinatarios.
- El servicio proporcionado deberá ser flexible, al objeto de acomodar la recogida de información de una amplia variedad de sensores de distinta naturaleza y capacidad.
- Se busca un sistema de bajo coste, por lo que su desarrollo se realizará a través de herramientas de código abierto.
- La plataforma será completamente configurable por parte del usuario a través de una interfaz web de gestión, que soportará la configuración flexible de los procesos de recogida, encapsulación y envío de los datos de sensores de los UAVs gestionados.

- El correcto funcionamiento del sistema implementado será verificado sobre una maqueta de pruebas de UAS, disponible en el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid.

1.3. Estructura del documento

El documento es dividido en siete capítulos y cinco anexos, los cuales se describen a continuación:

- El capítulo actual, **Introducción**, cubre aquellos aspectos relacionados con el contexto en el cual se desarrolla el proyecto, los objetivos a alcanzar y la descripción del la organización del contenido de la memoria.
- En el capítulo 2, **Estado del arte**, se realiza un recorrido por la historia de los UAV y se presentan los distintos tipos de UAV que existen. Además de esto, se describen las tecnologías actuales ligadas al desarrollo de aplicaciones web.
- El capítulo 3, **Análisis y diseño del sistema**, examina el escenario en el que el sistema debe funcionar. En primer lugar, se describe el esquema del modelo principal y a continuación, se enumeran los requisitos de cada uno de los módulos que integran el sistema.
- En el capítulo 4, **Implementación del software**, se ofrece una aproximación de la solución escogida. Después, se explican en detalle procedimientos y casos de uso principales.
- El capítulo 5, **Validación funcional**, lista las funcionalidades cubiertas por las aplicaciones implementadas. Además, a través de distintas gráficas, muestra los resultados de probar el funcionamiento del sistema.
- El capítulo 6, **Planificación del proyecto**, describe la organización del proyecto en su conjunto en términos de gestión del trabajo y análisis de costes.
- En el capítulo 7, **Conclusiones y líneas futuras** se comentan las dificultades encontradas durante el desarrollo del proyecto así como los resultados obtenidos. Además, se introducen futuras mejoras y líneas de trabajo.
- Anexo A, **Diagramas de clase y esquema XML**, incluye los diagramas de clase de las dos aplicaciones del sistema y la descripción de los principales elementos de configuración.

- En el Anexo B, **Descripción de la maqueta**, se detallan los principales componentes de la maqueta de pruebas.
- Anexo C, **Introducción**, incluye la traducción del capítulo de introducción en inglés.
- En el Anexo D, **Resumen extendido**, se presenta un resumen extendido en inglés de la memoria del proyecto.
- Anexo E, **Conclusiones y líneas futuras**, es la traducción del capítulo de conclusiones al inglés.

Capítulo 2

Estado del arte

En este capítulo se presenta el estado actual de la tecnología ligada al desarrollo de aplicaciones web en entornos de sistemas aéreos no tripulados. Con objeto de lograr una concepción completa del punto en el que se encuentra el mercado de UAV, se realiza un recorrido sobre la evolución cronológica de estos vehículos aéreos no tripulados en cuanto a sus capacidades y aplicaciones, además de introducir brevemente las herramientas software utilizadas hoy en día que permiten avanzar en este campo, así como los equipos hardware disponibles en el mercado para la recogida y procesamiento de datos obtenidos por los sensores de los UAVs. Por último se comenta brevemente el marco legal en el que se encuadra el uso de estos vehículos.

2.1. UAV

2.1.1. Concepto

El concepto UAV es el acrónimo más aceptado actualmente para hablar de los aviones que no necesitan un piloto a bordo para realizar una determinada misión.

Un UAV es básicamente un avión sin tripulación creado especialmente para ser dirigido desde una estación de control (GCS, en inglés *Ground Control Station*) y que lleva en su interior un sistema de comunicaciones capaz de recibir datos de la GCS y enviarlos de vuelta simultáneamente [5]. A este conjunto de sistemas se le conoce como UAS (*Unmanned Aircraft System*, sistema aéreo no tripulado) y es la denominación preferida por la industria y los reguladores [4], ya que hace alusión tanto al vehículo en sí como al resto de subsistemas cuyo despliegue es necesario para el control del avión.

No se debe confundir este concepto con el avión de aeromodelismo que debe estar dentro del campo de visión de la persona que posee el mando de radio control y que está limitado a una serie de movimientos, ni con aquellos aviones que pueden volar más allá de la vista del operador pero que no poseen ninguna inteligencia. Un UAV, o dron como se conoce coloquialmente, en cambio, es un sistema formado por múltiples subsistemas cuyo conjunto posee cierto grado de inteligencia de manera autónoma, siendo capaz de enviar los datos que recogen sus sensores así como información con el estado de los componentes del sistema; además, el UAV puede estar programado para solucionar fallos en los componentes por sí mismo y al tratarse de sistemas no estáticos, existe la posibilidad de poder controlar el UAV desde una o varias estaciones remotas y de que una sola estación controle varios UAV.

Por último, es importante hacer mención de la generalización que está comenzando a conseguir el acrónimo RPA (en español, "avión tripulado de forma remota"). Este concepto, que surgió con fuerza hace unos pocos años en EEUU para conseguir una aceptación social del uso de esos sistemas en medios urbanos, ha ido haciéndose hueco en todo tipo de informes de la UE para hablar de aviones no tripulados de uso civil.

En relación con esta memoria, es necesario mencionar que los términos RPA y UAV son utilizados indistintamente.

2.1.2. Evolución

El estilo futurista de su diseño y sus múltiples capacidades hacen que los UAV de hoy en día parezcan inventos irreales. Nada más lejos de la realidad, los sistemas que están viendo la luz en la actualidad son fruto de todo un siglo de inversión e investigación en el sector.

Las numerosas guerras que se han desatado a nivel mundial desde 1900 han supuesto la base del desarrollo de UAV; al ser necesario poseer un armamento superior tecnológicamente al del enemigo, los países más poderosos económicamente comenzaron a invertir en investigación, viéndose así el potencial que tenían estos sistemas no tripulados como elementos de espionaje y ataque.

La historia de los UAV [6] comienza a la mitad del siglo XIX, cuando los militares austriacos deciden lanzar globos cargados con explosivos, únicamente dirigidos por el viento, para atacar ciudades italianas. Será ya en la Primera Guerra Mundial (1914-1918), cuando se construya el Kettering Bug, el primer biplano que voló con una ruta preestablecida gracias a la incorporación de un altímetro y un giroscopio.

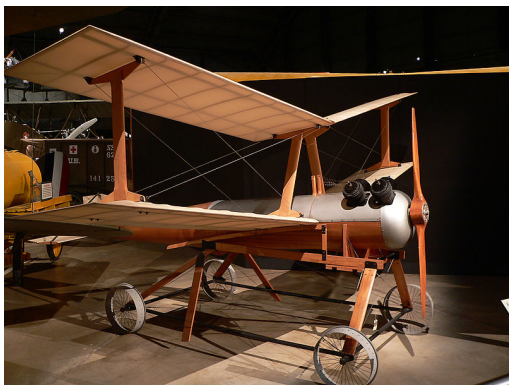

Figura 2.1: Biplano Kettering Aerial (EEUU)[1]

En la Segunda Guerra Mundial (1939-1945), la Alemania nazi construye un UAV llamado V-1, diseñado para volar más de 150 millas (alrededor de 240 km) antes de soltar los explosivos. Durante este segundo periodo bélico el avión controlado vía radio también evoluciona aunque es utilizado casi en exclusiva para destruir objetivos y no como sistemas de combate.

Figura 2.2: V-1 Flying Bomb (Alemania) [2]

Durante los años 70 se desata el conflicto de Yom-Kippur en Oriente Medio el cual supone un impulso definitivo y necesario para que los UAV alcancen la era moderna. Israel utilizó de manera coordinada cientos de aviones tripulados y no tripulados para combatir al frente árabe, convirtiéndose desde ese instante en uno de los desa-

[1]Fotografía del modelo Kettering Aerial Torpedo (EEUU)[Imagen]. Recuperado de http://commons.wikimedia.org/wiki/File:KetteringAerialTorpedo.jpg?uselang=es

[2]Fotografía del UAV V-1 (EEUU)[Imagen]. Recuperado de http://commons.wikimedia.org/wiki/File:V-1_flying_bomb_(3276223811).jpg?uselang=es

rrolladores de UAV más agresivos del último siglo. La Fuerza Aérea Israelí compra a Estados Unidos varios AQM-34 y los modifica para crear los Firebees 1241, UAV que fueron utilizados como señuelos y que llevaron a Israel a conseguir la victoria.

Figura 2.3: AQM-34 Ryan Firebee (EEUU) [3]

Desde la Guerra del Golfo y hasta la actualidad los aviones no tripulados han estado presentes en todos los conflictos armados existentes. El éxito conseguido por Israel sirvió como fuente de inspiración a Estados Unidos, que decidió utilizar modelos israelíes para sus propios propósitos; un ejemplo de ello es el intento de adaptar el denominado Pioneer para ser lanzado desde navíos. A pesar del esfuerzo por conseguir la adaptación, este UAV no disponía de mecanismos de despegue automático y las capacidades de aterrizaje y misión debían de ser controladas desde una GCS, lo que hacía imposible su uso desde un barco.

No obstante, el Pioneer y otros UAV israelíes han sido tomados como referencia por Estados Unidos para diseñar sus propias aeronaves y aún hoy en día se siguen utilizando UAV cuya base reside en estos modelos. Desde 1990 y hasta el día de hoy se han construido multitud de aviones no tripulados con fines que van más allá de objetivos militares; un buen ejemplo, es el UAV Helios, actualmente en desarrollo, construido con células de combustible que le dan la capacidad de mantenerse en vuelo durante un mes. Se espera que este aeroplano se convierta en diez años en una plataforma de comunicaciones de banda ancha utilizada por todo el mundo, es decir, un complemento rentable a los sistemas satélite-tierra ahora existentes.

[3]Fotografía del UAV de los años 60 AQM-34 Ryan Firebee (EEUU)[Imagen]. Recuperado de http://commons.wikimedia.org/wiki/File:Teledyne-Ryan-Firebee-hatzerim-1.jpg

[4]Imagen del Pioneer (Israrel) (1986) [Imagen]. Recuperado de http://commons.wikimedia.org/wiki/File:Pioneer_Unmanned_Aerial_Vehicle.jpg

[5]Fotografía del UAV Helios en vuelo. [Imagen]. Recuperado de http://commons.wikimedia.org/wiki/File:Helios_Prototype_on_Lakebed_-_GPN-2000-000198.jpg?uselang=es

Figura 2.4: Pioneer (Israel) [4]

Figura 2.5: Helios (UAV) [5]

2.1.3. Clasificación

Los vehículos aéreos no tripulados pueden ser clasificados siguiendo diversos criterios como puede ser el tipo de motor (explosión, eléctrico o turbojet) y los mecanismos de despegue (alas rotatorias para despegue vertical o alas fijas para despegue no vertical), pero la clasificación más común es la basada en el tamaño del vehículo. Según este criterio, existen numerosas plataformas de tamaño reducido (pequeñas, mini y micro) y otras de mayores dimensiones; el parámetro del que se está hablando es el peso máximo de despegue (en inglés MTOW, *Maximum Take-Off Weight*), que consiste en la suma del peso de la aeronave en vacío, la máxima carga (*payload*) transportable y el máximo combustible. Así, el tamaño estará directamente relacionado con la capacidad de vuelo, altura y alcance del

UAV.

En la siguiente figura se muestra la clasificación que hace la OTAN (Organización del Tratado del Atlántico Norte, en inglés: *North Atlantic Treaty Organization, NATO*) a este respecto, donde se observa que los UAV con mayor peso y autonomía de vuelo reciben la denominación MALE (*medium-altitude long-endurance*) y HALE (*high altitude, long endurance*).

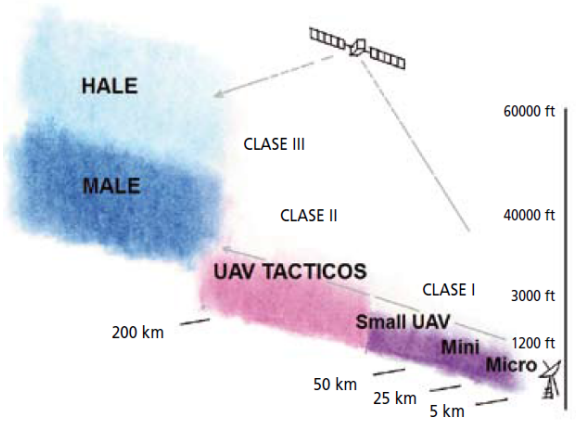

Figura 2.6: Clases de UAV según la clasificación de la OTAN. [47]

2.1.4. Usos y aplicaciones

Desde hace unos años, el sector de la aviónica no tripulada pretende ampliar su espectro de uso de manera que se vaya perdiendo esa asociación, hasta ahora bastante frecuente, entre los UAV y el mundo militar. Una de las ventajas con las que cuenta este tipo de sistema es su gran flexibilidad de uso; en la agricultura se utiliza como método de reconocimiento del terreno, así, se puede determinar qué zonas son más idóneas para los cultivos puesto que se tiene un conocimiento casi exhaustivo de cada parcela [34]. Servicios de emergencia en operaciones de búsqueda y rescate [29], ayuda humanitaria, detección de incendios y monitorización de la contaminación y el ruido [48] son otras de las aplicaciones que ya están funcionando en distintos lugares del mundo.

Debido al objeto de uso de la aplicación desarrollada en este proyecto, el foco de atención se centrará en los UAV de uso táctico y en los sistemas de tamaño reducido que serán introducidos en detalle más adelante.

UASs tácticos

Los conocidos como TUAV (*Tactical Unmanned Aerial Vehicle*, vehículo aéreo no tripulado de uso táctico), son aquellos UAV cuyo ámbito de actuación se encuentra en el rango entre los mini UAV de corto alcance y los potentes MALE (*medium-altitude long-endurance*) y HALE (*high altitude, long endurance*) UAV; estos sistemas combinan la flexibilidad de las pequeñas plataformas con la resistencia de los equipos más grandes. Los sistemas de este tipo están creados para operar en entornos hostiles y ser capaces de recoger, analizar y distribuir los datos que recogen de forma eficiente, precisa y segura [37], ejemplo de ello son los modelos que se presentan a continuación.

SIVA Este sofisticado sistema no tripulado de vigilancia aérea [21] tiene múltiples aplicaciones tanto en terreno militar como en uso civil, pudiendo ser utilizado como vehículo de observación en tiempo real. Desarrollado íntegramente en España bajo la supervisión del INTA (*Instituto Nacional de Técnica Aeroespacial*), el SIVA cuenta con un avión pilotado remotamente cuya autonomía es de siete horas, lo que le capacita para realizar misiones de manera completamente automática mientras es controlado y monitorizado desde la GCS.

Figura 2.7: SIVA (España) [6]

A efectos del presente proyecto, es necesario indicar que el software desarrollado ha sido verificado en un router que ya se ha embarcado en diferentes misiones del SIVA. Este router posee una herramienta de adquisición de datos o DAU (del inglés, *Data Acquisition*

[6]Ejército del Aire. (2013). [Imagen del SIVA UAV]. Recuperado de http://www.portierramaryaire.com/foro/viewtopic.php?f=5&p=171260

Unit, DAQ - Adquisición de Datos, en español) donde los sensores son conectados. Estos centros de obtención de datos, son los responsables de convertir las señales analógicas recibidas a través de los sensores en valores digitales. En la mercado actual, existen diversos equipos como son el MW100 [16] de la empresa Yokogawa Electric o el CompactDAQ [33] propiedad de National Instruments. El DAU instalado en el router del SIVA es un ACRA [18] caracterizado por ser compacto y tener un bajo consumo de potencia, lo que le hace ideal para lugares con espacio reducido como el interior de un UAV. Además, cuenta con una capa de rugerizado, lo que le permite operar en entornos hostiles.

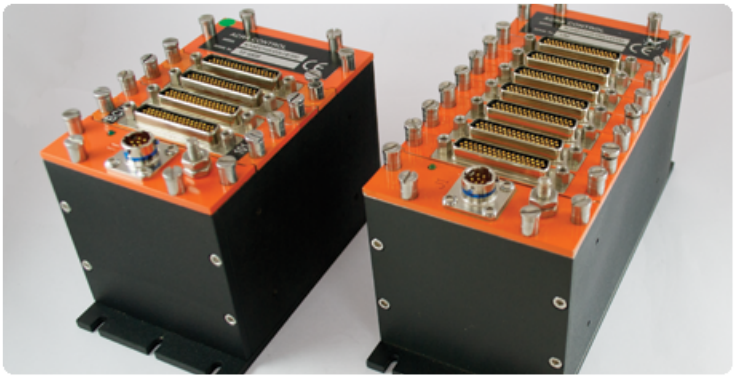

Figura 2.8: DAU Acra KAM-500

DIANA Diseñado como sistema de entrenamiento aéreo para posibles amenazas futuras, el DIANA es un UAV de blanco aéreo de altas prestaciones desarrollado por el INTA con ciertas mejoras con respecto otros modelos como son la simplificación de su fabricación, integración y mantenimiento.

Figura 2.9: DIANA (España) [7]

PREDATOR Desarrollado y construido actualmente por las Fuerzas Aéreas de los Estados Unidos, Predator está incluido dentro del top-ten de UAV que han revolucionado el mercado de la aviónica no tripulada. Este sistema de gran resistencia y fiabilidad [68] tiene como objetivo realizar misiones de inteligencia, reconocimiento y vigilancia a una altitud media, para ello, consta de numerosos sensores y cámaras infrarrojas de altas prestaciones con las que se consiguen imágenes de calidad que pueden ser distribuidas en tiempo real a cualquier punto del mundo a través de comunicaciones vía satélite. Recientemente, el fabricante de este potente UAV, General Atomics Aeronautical Systems (GA-ASI), y la empresa española de ingeniería y tecnología SENER, han alcanzado un acuerdo de colaboración que pone a disposición del Ministerio de Defensa del Gobierno de España el primer sistema operativo de este tipo en el país [20].

Figura 2.10: Predator UAV (EEUU) [8]

Micro y nano UAS

A finales de los años 90, DARPA (*Defense Advanced Research Projects Agency*, Agencia de Proyectos de Investigación Avanzados de Defensa) comienza dos proyectos que supondrían el nacimiento de los conceptos MAV (*Micro Air Vehicle*) y NAV (*Nano Air Vehicle*). Por MAV se entiende aquel sistema que puede estar en el aire durante una hora, cuya carga no puede ser superior a 5 kg y con capacidad para operar en un rango de 10 km. Un NAV, en cambio, está diseñado para poder soportar un peso de hasta 25 gr y operar durante menos de una hora en un rango de 1 km de distancia [41].

[7]Sistemas de Control Remoto S.L. (2011). [Imagen del DIANA UAV]. Recuperado de http://www.scrtargets.es/English/diana.html

[8]General Atomics Aeronautical. (2013). *Predator UAS* [fotografía] Recuperado de http://www.ga-asi.com/resources/library/var/resizes/Images/Aircraft-Platforms/Predator/P9.JPG

Con un coste económico y energético reducido en comparación con los UAV tácticos, estos pequeños vehículos aéreos se están convirtiendo en una herramienta perfecta para desarrollar aplicaciones orientadas a misiones colaborativas [44] tales como operaciones de rescate, vigilancia y refuerzo del orden público.

Otras utilidades de MAVs y NAVs son la generación de imágenes en situaciones de emergencia [46] y la creación de redes de sensores aéreas como ayuda de control en catástrofes [45]. Por otro lado, existen aplicaciones que requieren comunicación y coordinación entre UAV de distintos tamaños; un ejemplo de ello es el sistema que ha sido estudiado en [43] y desarrollado en [60] , donde se propone utilizar estos aviones no tripulados como medio de comunicación aérea dando soporte a una red ad–hoc en tierra, mejorando su fiabilidad y rendimiento[12].

Es importante comentar que aunque existen numerosos MAVs y NAVs ya comercializados, sus capacidades están aún por explotar. Son UAV cuyo sistema operativo es libre lo que da la posibilidad a los desarrolladores de crear sus propias aplicaciones sobre un SDK con software ya programado y probarlas en los aviones [27].

Figura 2.11: AR Drone 2.0 (Parrot) [9]

2.2. Aplicación web

Una aplicación web (*web-based application*) es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (*HyperText Transfer Protocol* (HTTP)) están estandarizados y no han de ser creados por el programador de aplicaciones [50].

El funcionamiento de una aplicación web se muestra en la figura 2.12 y es el siguiente: el navegador o cliente web realiza una petición

[9] AR Drone2 Parrot. (2014) Ejemplo de mini UAV comercial [fotografía] Recuperado de http://ardrone2.parrot.com/media/uploads/photos/album/hd/ardrone2_hd_outdoor_g.jpg

HTTP solicitando un recurso mediante su URI (*Uniform Resource Identifier*, identificador de recursos uniforme) , o generalmente su URL (*Uniform Resource Locator*, localizador de recursos uniforme), a un servidor web que los tiene almacenados y que envía el contenido de estos recursos en una respuesta HTTP al cliente web, que interpreta este contenido y lo presenta al usuario [50].

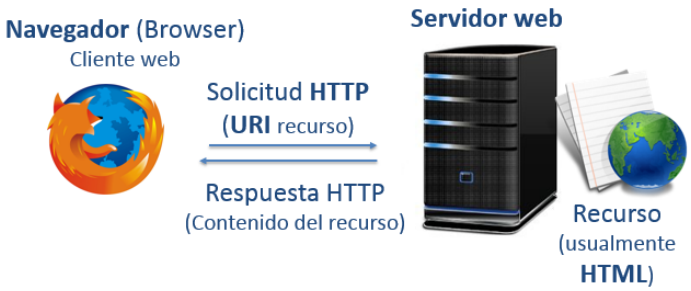

Figura 2.12: Funcionamiento de las aplicaciones web[10]

2.2.1. Arquitecturas de las aplicaciones web

Una vez presentados los componentes básicos de una aplicación software: servidor web, cliente web y conexión de red, es preciso hablar de las tres funcionalidades principales, éstas son, presentación de la información, procesamiento o lógica de negocio y administración de los datos. Dependiendo de cómo se repartan estas tres funciones, la arquitectura de la aplicación será de dos o más capas (sistemas *multi-tier* [67]).

Modelo de dos capas

En este modelo la información atraviesa dos capas entre la interfaz y la administración de los datos. La mayor parte del peso de la aplicación está en el lado del cliente (*fat client*) que soporta la lógica de negocio, mientras que el servidor únicamente se encarga de administrar los datos. Este modelo es bastante limitado ya que su flexibilidad es restringida y es difícilmente escalable.

Modelo de N-capas

En la búsqueda de una solución para superar las limitaciones del modelo de dos capas, aparece un nuevo diseño que introduce

[10]Fernández, Norberto (2013). Introducción a las aplicaciones web.

una capa intermedia con objetivo de manejar los procesos de forma separada a la interfaz de usuario y los datos [9]. Esta capa intermedia centraliza la lógica de negocio, simplificando así la administración de los datos, por ello, las aplicaciones web actuales se ajustan a este modelo.

2.2.2. Lenguajes de desarrollo web

Es en el lado del servidor donde se programa tanto la lógica de negocio de la aplicación como el flujo de interacción de ésta, es decir, de qué manera se organiza la navegación a través de páginas HTML (del inglés *HyperText Markup Language*, lenguaje de marcas de hipertexto). Para hacer realidad todo ello existen numerosos lenguajes de programación tales como Java, PHP, ASP.NET o Perl, mediante los cuales se implementan las funcionalidades que determinan qué puede hacer el usuario gracias a la aplicación. Echando un vistazo al ranking que se presenta en la siguiente figura, se puede ver que son PHP, ASP.NET y Java los lenguajes de programación que más importancia tienen actualmente en cuanto al desarrollo de aplicaciones web. En las próximas secciones se profundiza en aspectos más concretos de PHP y Java, dejando de lado a ASP.NET por ser una tecnología compatible únicamente con el sistema operativo Windows de Microsoft [49].

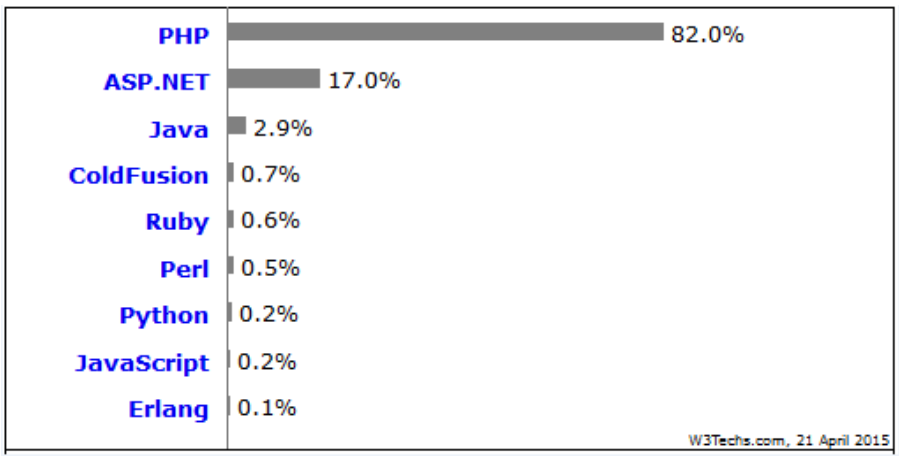

Figura 2.13: Ranking de lenguajes de programación en el lado del servidor [11]

[11] Porcentaje de uso de distintos lenguajes de programación del lado del servidor en sitios web. (2015) [gráfico] Recuperado de http://w3techs.com/technologies/overview/programming_language/all

Java

La plataforma creada por Sun Microsystems, Java EE (*Java Platform Enterprise Edition*) [70], para el desarrollo de aplicaciones empresariales con Java, incluye la tecnología JSP (*Java Server Pages*) para crear páginas web dinámicas en el servidor de manera que puedan acceder a una base de datos y exista interactividad con el usuario [35]. Una de las ventajas que ofrece JSP frente a otros lenguajes es que permite integrarse con clases Java lo que hace posible la separación en niveles de la aplicación web, pudiendo así almacenar en clases Java las partes que requieren más seguridad y consumen más recursos, y alojando en el jsp todo lo que esté dedicado a formatear el documento HTML [13]. Por otra parte, Java es un lenguaje que puede ser ejecutado en cualquier sistema, lo cual unido al uso de jsp da una gran versatilidad a la aplicación.

PHP

PHP [7], acrónimo de *Hypertext Preprocessor*, es un lenguaje de programación interpretado con gran capacidad para mezclarse con el código HTML, lo que le convierte en una de las tecnologías más utilizadas para el desarrollo de páginas web activas. Utiliza un servidor gratuito e independiente de plataforma bajo la licencia GNU, aunque también existe un compilador comercial denominado Zend Optimizer [72], que garantiza el respaldo de este lenguaje en el mercado. Todo ello unido a una gran librería de funciones y su compatibilidad con todos los navegadores han elevado la popularidad de PHP notablemente [66].

2.2.3. Herramientas de diseño web

En el lado del cliente la única cuestión que se presenta es la elección del navegador web, ya que la opción elegida (Firefox, Chrome, Internet Explorer, etc.) delimita el conjunto de tecnologías software disponibles como son HTML, CSS (*Cascading Style Sheets, hojas de estilo en cascada*) [40], Javascript [32] o AJAX (*Asynchronous JavaScript And XML*) [30], entre otras. Estos lenguajes bien conocidos son directamente interpretados por el navegador sin necesidad de un tratamiento previo, el cual se encargará asimismo de dar el formato adecuado al contenido de la página. A continuación se presentan brevemente algunas de las tecnologías más comunes utilizadas en el lado del cliente.

HTML HTML es una recomendación del W3C (*World Wide Web Consortium*) para publicar documentos con texto, imágenes, tablas, etc. y acceder a otros recursos enlazados. Asimismo, permite intercambiar información a través de formularios para llevar a cabo tareas e incluir vídeo, audio u otros objetos en los documentos [25]. Se basa en etiquetas de marcado que describen cada parte del contenido del documento, lo cual ofrece una estructuración lógica y de fácil interpretación puesto que sigue un esquema de árbol donde existe un elemento raíz, insertándose el resto de etiquetas de manera estructurada y lógica.

CSS Las hojas de estilo en cascada nacen para facilitar el mantenimiento de HTML, separando presentación de contenido. Actualmente el W3C está trabajando en el Borrador de Trabajo de CSS3 [14] que permite la modularización, entre otras mejoras. Con CSS la información del estilo puede ser definida en un documento por separado o en el mismo documento que el código HTML, lo que permite un mantenimiento fácil de la web.

Javascript Mientras que HTML y CSS permiten hacer webs estáticas, el lenguaje de programación Javascript aporta interactividad a las mismas: cuando ocurre un evento en la página web (ej.: click) se ejecuta una función que manipula la información del documento. En la actualidad existen multitud de librerías que simplifican el desarrollo de Javascript como por ejemplo JQuery, una de las más potentes y utilizadas hoy en día [11] ya que proporciona soporte multinavegador y es fácilmente extensible puesto que existen multitud de plugins que amplían las funcionalidades que este framework ofrece [51].

AJAX AJAX no es una tecnología sino un conjunto de ellas acopladas de forma tal que se beneficia de las ventajas que aportan cada una de ellas [30]. Javascript, XMLHttpRequest, XML, XHTML y CSS son combinadas con el objetivo de mejorar el tiempo de respuesta y reducir recursos cargando sólo los datos que necesitamos del servidor, los cuales serán utilizados para modificar el código HTML mediante Javascript. Para que el usuario no perciba este proceso AJAX se sirve de comunicaciones asíncronas mediante el objeto XMLHttpRequest del navegador. En la siguiente figura se muestran las diferencias entre el funcionamiento tradicional de las webs y el diseño de una aplicación con AJAX.

[13]Garrett, J. J. (2005). Ajax: A new approach to web applications.

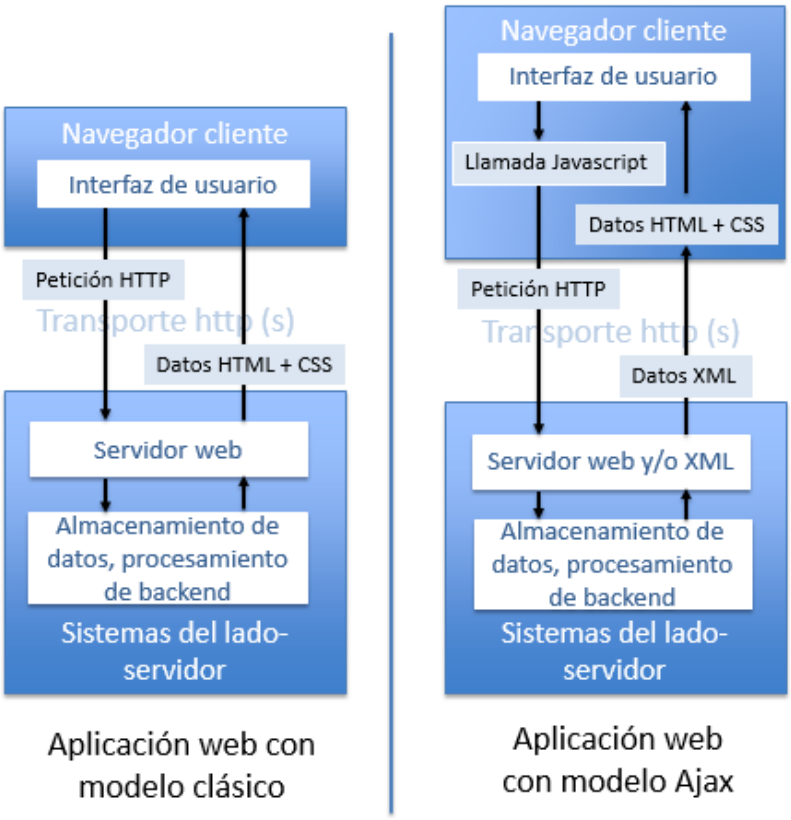

Figura 2.14: Modelo tradicional de aplicaciones web (izquierda) comparado con el modelo AJAX (derecha) [13]

2.2.4. Gestión de la información

Las aplicaciones Web requieren almacenamiento y tratamiento de datos para su correcto funcionamiento. En la década de los setenta, antes de que aparecieran los Sistemas Gestores de Bases de Datos (SGBD, en inglés *Database Maganement System, DMS*), la información se trataba y gestionaba utilizando sistemas de gestión de archivos donde el cambio de un dato en un archivo suponía la modificación del resto de programas dentro del sistema. Para superar los problemas que el uso de esta opción traía consigo aparece el SGBD, consistente en un conjunto de ficheros que almacenan datos y un software que facilita el acceso a dichos ficheros. Las ventajas de utilizar estos sistemas gestores son, entre otras, la flexibilidad e independencia entre aplicaciones, la integridad de los datos y la disminución del tiempo de desarrollo de las aplicaciones [24][3].

Fundamentalmente, existen cuatro tipos de SGBD: jerárquicos,

de red, orientados a objetos y relacionales. En la actualidad, son estos últimos los más utilizados debido especialmente a su facilidad de manejo; en este modelo, los datos se describen como relaciones que se suelen representar como tablas bidimensionales consistentes en filas y columnas, donde cada fila (tupla) es una ocurrencia y cada columna (atributo), una propiedad. Otro de los motivos a los que debe su fama el SGBD relacional, es la implantación del lenguaje SQL (del inglés, *Structured Query Languaje*, lenguaje de consulta estructurado) [63] como estándar para el manejo de datos en este modelo.

Uno de los problemas que supone el uso de SGBD relacionales es la dificultad de adaptar el uso de objetos a tablas de una base de datos relacional, aunque existen tecnologías como Hibernate [31] que facilitan este trabajo, una buena alternativa es utilizar XML y herramientas de enlace (*binding*) para serializar las referencias de objetos Java a XML y viceversa.

XML

El lenguaje de etiquetado extensible o XML (*eXtensible Markup Languaje*) es una tecnología muy simple y similar a HTML, cuya función principal es describir datos en lugar de mostrarlos como es el caso de HTML. Desarrollado por el W3C, XML deriva del lenguaje SGML (*Standard Generalized Markup Language* o Lenguaje de Marcado Generalizado Estándar) y da soporte a bases de datos, lo que permite que varias aplicaciones en diferentes plataformas puedan comunicarse entre sí e integrar información de manera segura, fiable y fácil [23].

Alrededor de XML existen distintas tecnologías (XSL, XPath, XLink...) que la complementan y permiten cubrir mayor número de necesidades de los usuarios. Todo ello ha hecho que XML tenga un papel muy importante en la actualidad.

Procesado de XML en Java

La información de un documento XML se puede representar como un objeto en la memoria del ordenador, de manera que una aplicación puede acceder a los datos en el XML desde el objeto. En concreto, se crea una correspondencia entre los elementos del esquema del XML (*XML Schema*) y los objetos en memoria que permite convertir un documento XML en objeto y al contrario, serializar un objeto como XML [10].

A continuación se presentan dos tecnologías capaces de realizar el *binding* a alto nivel.

JAXB *Java Architecture for XML Binding* (JAXB) o Arquitectura de API Java para Uniones XML [57], permite almacenar y recuperar datos en memoria en cualquier formato XML.

Dado un esquema XML, el compilador JAXB genera un conjunto de clases Java que contienen el código necesario para que una aplicación pueda manipular el árbol de objetos. Es particularmente útil cuando la especificación es compleja y cambiante ya que cambiar regularmente el esquema XML para mantener la sincronización con los objetos Java suele dar lugar a errores.

StAX StAX (*Streaming API for XML*), es una tecnología creada por la comunidad del lenguaje de programación Java para servir de término medio entre las APIs XML basadas en DOM (*Document Object Model* o Modelo de Objetos del Documento) [69] como JAXB y las basadas en eventos como SAX (*Simple API for XML*) [56]; su funcionamiento se basa en "empujar" los datos hacia la aplicación según ésta la vaya necesitando. StAX puede leer o escribir XML, acceder aleatoriamente a cualquier punto del documento y solo requiere una pequeña huella de memoria, lo que la hace más rápida que las APIs que cargan en memoria el árbol de objetos completo.

2.3. Marco regulatorio

En Julio del año 2014 el Consejo de Ministros de España aprueba el Real Decreto-Ley 8/2014. Esta ley crea un marco regulatorio temporal que establece unos requisitos exigidos a las aeronaves no tripuladas en términos de frecuencia de transmisión de drones y tamaño de los mismos, entre otros aspectos, para poder operar a nivel nacional y europeo.

Sin embargo, en el decreto no se expresa especificación alguna que restrinja los mecanismos de encapsulación y envío de datos de telemetría (sensores). Por ello, debido a la ausencia de limitaciones legales en este aspecto, en el presente proyecto se va a explotar una plataforma que permita configurar ambos mecanismos de manera flexible.

Capítulo 3

Análisis y diseño del sistema

Este capítulo recoge los requisitos que debe cumplir la aplicación. Para comprender el contexto que los engloba, se realiza previamente un descripción del problema cuya solución vendrá dada por el sistema que en este proyecto se propone.

3.1. Escenario del problema

El escenario en el que se trabaja consiste en el típico sistema de RPA donde el modelo inicial está inspirado en un RPAS táctico de vigilancia ya existente, el SIVA [21], creado por el Instituto Nacional de Tecnología Aeroespacial (INTA) y financiado por el Ministerio de Defensa de España.

El esquema consiste en un conjunto de unidades RPA de diferentes alcances (tácticos, MALE o HALE) y capacidades, los cuales están subordinados a una GCS. Cada RPA puede llevar distintas cargas, dependiendo de la misión que tenga que realizar el sistema completo; así pues, un RPA puede tener integrados un sensor de temperatura, cámaras de video o infrarrojas, radares de apertura sintética, sensores RFID (*Radio Frecuency Identification* o identificación por radiofrecuencia) u otro tipo de equipamiento electrónico. La información generada por cada sensor o dispositivo en los distintos RPA es recogida y procesada por una unidad de adquisición de datos que la envía como telemetría a la GCS desde donde se retransmite hacia diferentes equipos encargados de analizar dichos datos. De igual forma, se pueden enviar información por telecomando desde la GCS hacia las unidades RPA para cambiar la ruta establecida o la orientación de la cámara de vídeo, por ejemplo.

Las comunicaciones sin cable entre la GCS y cada RPA se realizan mediante enlaces con línea de visión (en inglés, LOS, *Line-of-sight*); para prevenir pérdidas de comunicación debido a posibles fallos de conectividad, el modelo soporta la existencia de enlaces de backup vía satélite.

En la siguiente figura se muestra el modelo descrito anteriormente, de manera que se puede ver cómo están repartidos y vinculados los distintos elementos que lo componen.

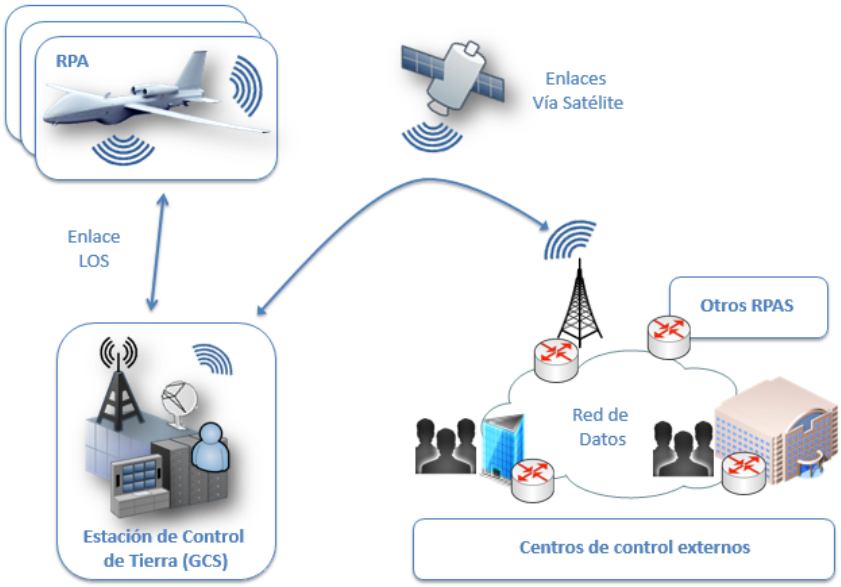

Figura 3.1: Visión de conjunto de la arquitectura del sistema

Sobre este modelo se quiere desarrollar un sistema de gestión de la información proveniente de los sensores alojados en los RPA, de manera que desde los equipos conectados a la GCS, o desde redes externas, se puedan configurar estos dispositivos y recibir los datos sin necesidad de que estos sean procesados en cada equipo de destino. La plataforma propuesta se divide en tres componentes distintos, módulo RPA, módulo servidor y módulo de acceso, relacionados entre sí tal y como se ve en la figura 3.2.

Una vez definido el escenario global en el que se va a desarrollar la aplicación y a modo de esclarecer la funcionalidad de cada módulo que integra el sistema, a continuación, se comenta brevemente el diseño de cada uno y se realiza un análisis de los requisitos que deben cumplir para conseguir los objetivos propuestos.

3.2. Módulo A: RPA

El RPA lleva instalados los sensores que recogen la información del medio. Para poder procesar y analizar dicha información, el sistema lleva integrado una aplicación que recoge estos datos, los encapsula creando flujos[1] y los reenvía a aquellos usuarios que hayan solicitado previamente la información. Esta aplicación se actualiza con cada cambio en la configuración del sistema.

3.2.1. Requisitos del módulo A

Para llevar a cabo el desarrollo del módulo anteriormente descrito se han establecido una serie de requisitos, que son:

- La aplicación debe utilizar tecnología de almacenamiento de datos ligera sin necesidad de recurrir a bases de datos.
- La comunicación con el servidor, al efecto de configurar los procesos de recogida, encapsulación y envío de información, debe realizarse garantizando la integridad y confidencialidad de la información intercambiada.
- La estructura de la aplicación debe ser fácilmente extensible y modificable.
- La aplicación debe permitir la configuración de un número arbitrario de sensores en un RPA.
- La aplicación debe permitir enviar cada flujo de datos con una periodicidad distinta.
- La aplicación debe comprobar la marca de tiempo de los datos que recibe de los sensores para evitar el envío de información obsoleta.
- La aplicación debe permitir recuperar de los distintos sensores datos válidos, es decir que cumplan con un patrón definido por el usuario.

3.3. Módulo B: servidor de datos

Este módulo es el encargado de enlazar los otros dos elementos del sistema, es decir, realiza una función de pasarela entre el RPA y el usuario, comunicando al primero los cambios que el segundo quiere

[1] Se entiende por flujo aquel conjunto de datos proveniente de uno o más sensores.

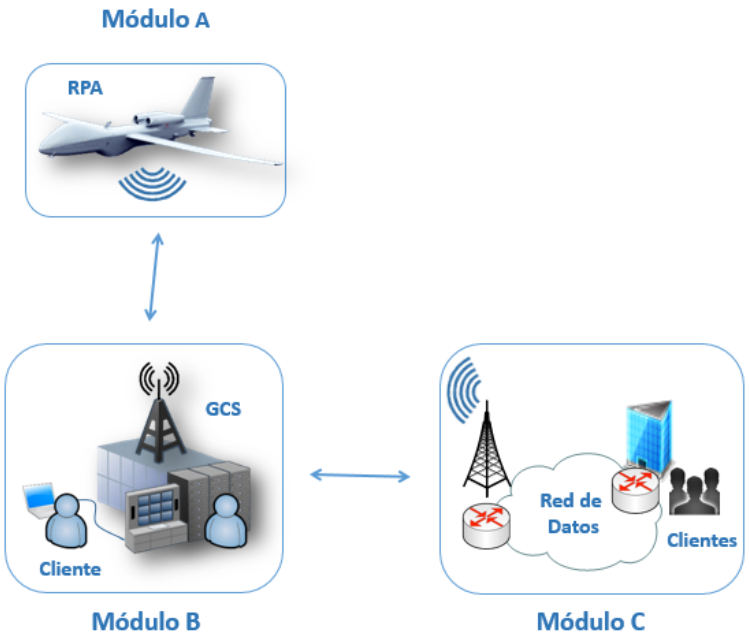

Figura 3.2: Esquema del sistema representado en módulos

efectuar en los sensores y en los flujos de información, y reenviando los datos requeridos desde el RPA a los equipos donde los usuarios desean recibirlos. Esta funcionalidad se representa en la Figura 3.2, donde se muestra cómo se dan las relaciones entre los elementos del esquema propuesto.

3.3.1. Requisitos del módulo B

Las especificaciones que debe cumplir el servidor son detalladas a continuación:

- El servidor soportará la configuración en cada RPA de las operaciones de recogida, encapsulación, y envío de datos de los sensores, atendiendo a la información indicada por el usuario a través del módulo C, con la que el servidor mantendrá una interfaz HTTP.
- La conexión entre el cliente y el servidor ha de ser segura.
- El servidor debe administrar los datos de configuración introducidos por el usuario sin hacer uso de bases de datos.
- El servidor debe contar con una herramienta de control de sesiones que permitan la autenticación del usuario con el RPA.

- El servidor debe garantizar la integridad y confidencialidad de los datos enviados al RPA.

3.4. Módulo C: acceso al usuario

Para que el usuario pueda comunicarse con los RPA es necesario que exista una interfaz a través de la cual este sea capaz de dar de alta o baja así como modificar sensores en la aplicación, además de configurar la recogida de datos de estos y la encapsulación y entrega de información mediante flujos. Cada sensor puede ser modificado sin afectar a la configuración del resto. Asimismo, cada flujo de información tiene unas características específicas e independientes de los demás flujos lo que garantiza la flexibilidad del sistema.

3.4.1. Requisitos del módulo C

Para concluir, se listan los requisitos referentes al módulo que permite al usuario acceder al sistema:

- El usuario debe poder acceder a la configuración del RPA mediante una plataforma Web.
- La plataforma Web deber ser plurilingüe.
- La interfaz debe mostrar una página de acceso al RPA mediante el par usuario/contraseña.
- La interfaz debe permitir al usuario elegir de qué sensores desea recibir información.
- La interfaz debe mostrar la hora y fecha actuales.
- La interfaz debe ser intuitiva y de manejo sencillo.
- La interfaz debe tener las opciones de añadir/eliminar/modificar sensores y flujos.
- La interfaz no debe permitir transiciones a estados inconsistentes (por ejemplo no debe permitir eliminar un sensor si hay al menos un flujo de datos solicitando información de este).
- La interfaz debe permitir, para cada sensor, establecer una sintaxis específica del tipo de datos que se quiere recibir.

Capítulo 4

Implementación del software

Este capítulo se dedica a la descripción del desarrollo de los módulos previamente diseñados. En concreto, se diferencia la implementación de dos aplicaciones distintas, la instalada en el RPA que se encarga de la gestión de los sensores, y la aplicación integrada en la GCS, diseñada para comunicar al usuario con el RPA.

4.1. Panorámica general de los módulos principales

Como se comenta en la introducción del capítulo, el sistema de gestión de sensores se basa en dos aplicaciones independientes compuestas por tres módulos distintos: módulo de acceso al usuario, módulo servidor o pasarela y módulo RPA. En la figura 4.1 se muestra cómo están relacionados estos elementos.

Los dos primeros módulos que se explican a continuación forman parte del denominado modelo Modelo-Vista-Controlador (MVC) [38], una arquitectura que permite la integración de las tecnologías principales de ambos módulos. En este patrón MVC cada componente tiene un rol:

- **Modelo**: consiste en el acceso y manipulación de datos. En el caso de esta aplicación, este rol lo desempeña la Entidad gestora, componente del módulo Servidor.
- **Vista**: es la interfaz de usuario. Se explica en el módulo de Acceso al usuario, donde se detallan las tecnologías ligadas a la implementación de la interfaz.

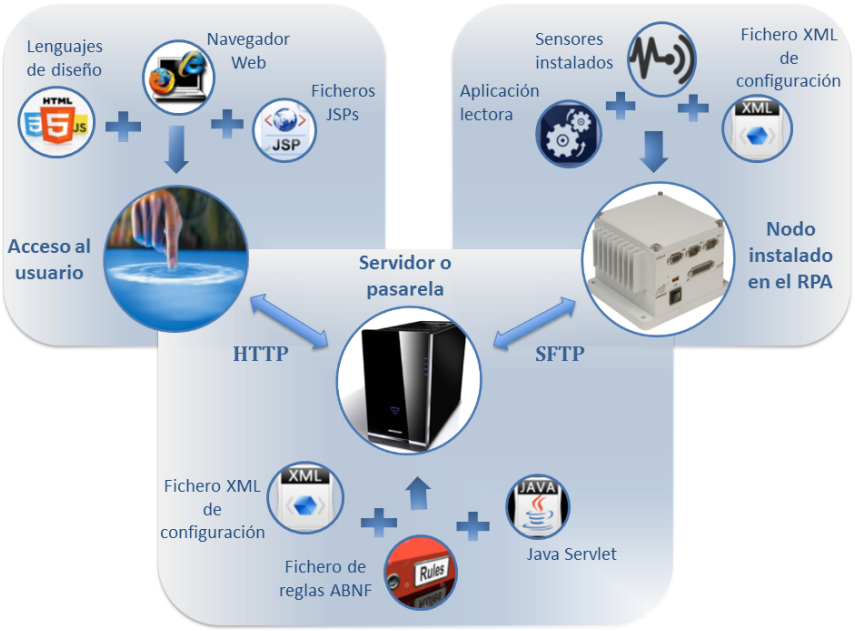

Figura 4.1: Relación entre los módulos del sistema

- **Controlador**: articula los otros dos elementos de la arquitectura; envía la información recibida en la solicitud a través de las vistas a componentes auxiliares que ejecutan la lógica de negocio y recibe el resultado de dicha ejecución. El Servidor Web, componente del módulo Servidor, es el encargado de llevar a cabo esta labor.

Con esta estructura se consigue desacoplar la vista y el modelo puesto que la vista no puede acceder directamente a los datos; así, se facilita el mantenimiento de la aplicación ya que se puede actualizar el modelo sin que el usuario tenga constancia de ello del mismo modo que se puede cambiar la vista manteniendo el modelo.

4.1.1. Módulo de acceso al usuario

El acceso a la aplicación por parte del usuario se garantiza a través de una interfaz gráfica que ofrece diversas funcionalidades como la autenticación del usuario, elección del idioma o la gestión íntegra de sensores y flujos de datos; los procedimientos llevados a cabo para realizar cada una de estas funciones se comentarán más adelante.

Esta interfaz es accesible a través de un cliente web o navegador, que solicita las páginas al módulo servidor para mostrarlas al usuario

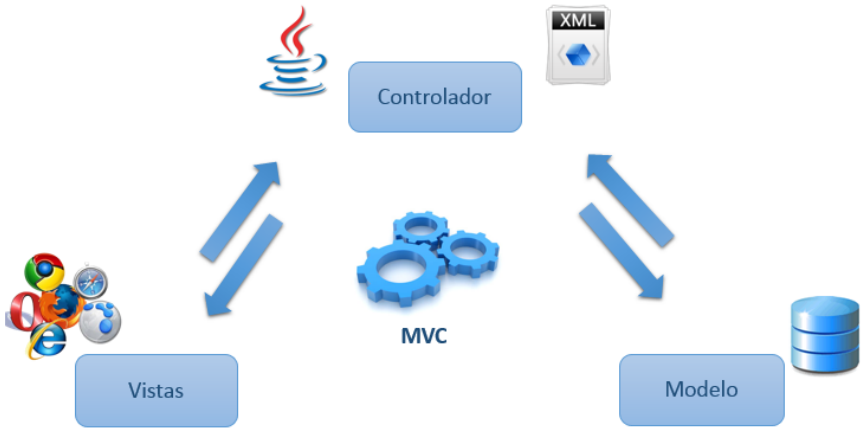

Figura 4.2: Esquema del patrón MVC

y permite que éste interactúe con el contenido de la página. Este módulo se comunica con el servidor a través del protocolo HTTPS mediante el esquema solicitud-respuesta.

Tecnologías software utilizadas

HTTPS Para comunicar el módulo de acceso con el módulo servidor, se hace uso de HTTPS o *Hypertext Transfer Protocol Secure* [58] (en español, Protocolo seguro de transferencia de hipertexto), un protocolo que trabaja a nivel de aplicación para garantizar la transferencia segura de datos de Hipertexto, esto es, la versión de HTTP sobre TLS (del inglés *Transport Layer Security*, seguridad de la capa de transporte) [22]. Mediante HTTPS se garantiza el envío cifrado de las credenciales de usuario una vez éste las ha introducido a través de la interfaz.

HTML Este lenguaje de programación es el elegido para estructurar el contenido de las páginas web que componen la interfaz de usuario. Gracias a su estandarización, cualquier navegador actualizado puede interpretar el código de la página para unir sus elementos y visualizarla.

```
<h1>APLICACIÓN DE CONFIGURACIÓN DE SENSORES</h1>
```

Figura 4.3: Ejemplo de elemento HTML

CSS La presentación de cada página web se ha determinado mediante una hoja de estilo en cascada, la cual permite definir para cada elemento o bloque de elementos HTML un conjunto de reglas de estilo. Actualmente, el aspecto gráfico de la web está optimizado para navegadores Mozilla Firefox y Google Chrome en sus versiones 33.0 y 42.0 respectivamente.

```
div{
    font-family: Arial, Helvetica, sans-serif;
}
```

Figura 4.4: Ejemplo de aplicación de estilo con CSS

JQuery La librería de Javascript, JQuery, se utiliza para permitir la interacción entre el usuario y la aplicación. Es una manera sencilla de acceder y manipular los elementos que forman la página web, además de facilitar el desarrollo de animaciones y efectos en la misma.

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
```

Figura 4.5: Ejemplo de acceso y manipulación de elementos con JQuery

JSP Es la tenología elegida para crear las vistas de la aplicación, permite integrar JQuery, CSS y HTML además de incrustar código Java en los documentos HTML para poder mostar la información que ha enviado el servidor en su respuesta. Esto se hace posible a través de directivas del tipo mostrado en la figura 4.6

```
<%@include file="header.jsp" %>
```

Figura 4.6: Ejemplo de código Java incrustado mediante JSP

Componentes del módulo de acceso

Descriptor de despliegue El fichero *web.xml* describe cómo se despliega la aplicación web. Es el encargado de asociar una URL o un patrón de URL con el módulo Java correspondiente.

JSPs Con objetivo de dotar a la aplicación web de distintas funcionalidades se han creado varias paginas JSP, las cuales se describen a continuación:

- *header.jsp*: es la cabecera de la aplicación web, contiene el título, el reloj y la fecha actual. Esta página se incluye en el resto de JSPs.
- *main.jsp*: es la página de inicio de la aplicación, permite elegir el idioma.
- *form.jsp*: contiene el formulario de autenticación a la aplicación. Presenta dos desplegables, uno para elegir el protocolo de transporte para el envío de la información de los sensores y otro para seleccionar el RPA al que el usuario se quiere conectar. En la actualidad, solo está soportado el protocolo SSH para conectar el servidor con el RPA.
- *sensors.jsp*: mediante esta página el usuario puede ver los sensores ya configurados en el RPA que ha seleccionado previamente (en el caso de que los haya), los cuales puede modificar o eliminar; además, tiene la posibilidad de añadir nuevos sensores. Desde esta página el usuario puede volver a la página inicial para conectarse a un nuevo RPA.
- *flows.jsp*: esta página, cuyas funcionalidades son muy parecidas a las que ofrece el JSP anterior, muestra al usuario los flujos ya establecido y le da la posibilidad de eliminarlos o modificar alguno de sus campos de configuración. Asimismo, permite añadir nuevos flujos para recibir información de los sensores pertinentes y volver a la página inicial para iniciar sesión con otro RPA.

Hoja de estilo La presentación de los elementos de todas estas páginas web dinámicas se realiza mediante una única hoja de estilo denominada *styles.css*. Este archivo se referencia desde todas los JSPs de la aplicación para mostrar su contenido con un determinado estilo de forma que se facilita el mantenimiento de la aplicación puesto que para modificar distintos elementos únicamente es necesario acceder a un archivo CSS.

4.1.2. Módulo servidor o pasarela

La conexión entre el usuario y el RPA está a cargo de este módulo servidor. Su existencia es fundamental, pues por una parte es el encargado de servir las páginas web al cliente desde el cual navega el usuario y por otra, vuelca los datos introducidos por éste a través de la interfaz en un documento XML. La comunicación con el RPA desde este módulo se realiza a través del protocolo SFTP que transmite el archivo XML con la información almacenada de forma segura mediante SSH[1] [39].

En el caso de que ya exista la configuración anterior de un sensor y/o un flujo en el RPA, ésta debe ser mostrada al usuario cuando se conecta al avión. Para ello se sigue el procedimiento inverso al descrito anteriormente, es decir, el RPA envía una copia local del XML con la configuración de sus sensores y flujos a través de SFTP al servidor, que traduce el XML en objetos Java que son manipulados y enviados a la interfaz donde son mostrados al usuario.

En las siguientes secciones se explica en detalle las tecnologías utilizadas para lograr el correcto funcionamiento de este módulo pasarela así como sus componentes.

Tecnologías software utilizadas

HTTPS Este protocolo que ha sido analizado en el módulo anterior, es el que comunica al usuario con el servidor a través de conexiones seguras, en concreto, el cliente realiza solicitudes sobre este protocolo que llegan al servidor, el cual las acepta, procesa y devuelve la respuesta pertinente.

ABNF En el esfuerzo por encontrar tecnologías que doten de flexibilidad al conjunto del sistema en cuanto a la configuración de sus elementos principales (flujos y sensores), se hace uso de la tecnología ABNF ó *Argumented Backus-Naur Form* [17]. Esta gramática o metalenguaje es utilizada comúnmente para describir protocolos IETF [52], ya que permite describir un sistema formal mediante una sintaxis y reglas de derivación propias. En concreto, una especificación ABNF consiste en una secuencia de símbolos que definen la regla, un comentario destinado a la documentación y concluye con un retorno de carro.

En la descripción de los componentes del módulo se presenta un ejemplo sencillo utilizando esta tecnología para facilitar la comprensión de su funcionamiento.

[1] La autenticación por SSH entre servidor y RPA se realiza mediante clave pública RSA

SFTP La transmisión de archivos entre distintos equipos, en este caso particular, entre el servidor y cada uno de las unidades RPA, se realiza a través de SFTP (del inglés, SSH File Transfer Protocol ó Secure File Transfer Protocol). Este protocolo de red provee acceso, transferencia y gestión de ficheros a través de un flujo de datos fiable, en definitiva, se trata de la versión 2.0 del protocolo SSH [71] pues fue diseñado por el IETF para añadir capacidades de seguridad a dicho protocolo.

Java En el estado del arte se comentan varios aspectos positivos de esta tecnología en relación con el desarrollo de aplicaciones web como es la capacidad de integración de este lenguaje con JSP; para que JSP pueda explotar sus funcionalidades es necesaria la existencia de Servlets en el servidor. Un servlet es una clase Java que hereda de la clase HttpServlet y permite gestionar elementos HTTP mediante las siguientes clases:

- *HttpServletRequest*: recibe la petición.
- *HttpServletResponse*: genera la respuesta.
- *HttpSession*: permite crear una sesión común a un conjunto de peticiones.
- *ServletContext*: gestiona la información común a todas las peticiones realizadas.

Por todo ello, el uso de esta tecnología se hace indispensable para garantizar el correcto funcionamiento de la aplicación que se quiere desarrollar, pues es la encargada de gestionar el enlace entre cliente y servidor a través de HTTP.

Además de la utilidad anterior, Java es la herramienta que permite a la aplicación comunicarse por SFTP con el RPA, así como gestionar la información de configuración del mismo mediante documentos XML a través de sus librerías:

- *Java Architecture for XML Binding* (JAXB): permite asignar clases Java a representaciones XML mediante la serializacion (Java a XML) y la deserialización (XML a Java).
- *Java Secure Channel* (JSch): Jsch es una implementación de SSH2 puramente en Java que pone a disposición de la aplicación todas las funcionalidades que ofrece SFTP como protocolo.

En futuros apartados se comenta cómo se integran todas estas tecnologías en el módulo servidor y cuál es su objeto de uso concreto.

Figura 4.7: Logotipo del Proyecto JAXB

XML XML (eXtensible Markup Languaje) o Lenguaje de Etiquetado Extensible [23], es una herramienta que se utiliza para almacenar datos de manera legible, permitiendo definir una gramática específica para estructurar documentos grandes. Actualmente, esta tecnología se propone como estándar para el intercambio de información estructurada entre plataformas distintas.

Alrededor de XML existen otras herramientas que la complementan y hacen aún mayor su alcance y sus capacidades como por ejemplo XML Schema, una tecnología utilizada para definir la estructura y restricciones de contenido de los documentos XML de una forma muy precisa, consiguiendo un mayor nivel de abstracción en la descripción del tipo de documento que se está generando. El XML Schema es similar al DTD, es decir, define qué elementos puede contener un documento XML, cómo están organizados así como el tipo de atributos de dichos elementos. A pesar de las similitudes entre ambas herramientas, XML Schema presenta importantes ventajas respecto a su competidor como la capacidad de especificar el tipo de datos y utilizar la sintaxis básica de XML, además de ser extensible.

Componentes del módulo servidor

Analizador sintáctico de gramática ABNF La configuración de cada sensor instalado en el RPA se realiza mediante la especificación de una serie de elementos; uno de estos es denominado *Ïnfo type"* y se utiliza para definir la sintaxis que siguen las lecturas del sensor. Para comprobar si la información recibida por el sensor cumple con la sintaxis y puede ser almacenada como datos que después serán enviados, es necesario construir una regla por medio de la cual se validen las lecturas realizadas.

Para poder llevar esto acabo, se hace uso de la herramienta **parse2**, que permite el uso de ABNF en Java a través de sus librerías de código libre.

Con objetivo de entender mejor el funcionamiento de este módulo, se presenta un sencillo ejemplo de un reloj, en el cual se define

la sintaxis de la lectura de las hora y se comprueba su validez:

1. Primero se define la sintaxis en un fichero ABNF al que llamamos, por ejemplo, clock.abnf y en el que se introduce:

```
Clock = Hours Separator Minutes [Separator Seconds]; #Seconds optional.

Hours = (%x30-31 %x30-39) / (%x32 %x30-33);

Minutes = %x30-35 %x30-39;

Seconds = %x30-35 %x30-39;

Separator = ":";
```

Figura 4.8: Definición ABNF de un reloj de 24 horas

2. Para poder aceptar su validez es necesario generar el parseador que comprueba las lecturas; el código fuente del parseador se genera mediante la siguiente sentencia:
   java -cp aparse.jar com.parse2.aparse.Parser clock.abnf
   Si la gramática de la regla es correcta, se crean varias clases Java. A pesar de que todas las clases son necesarias para que se pueda comprobar la validez de las lecturas, solo una de ellas va a ser instanciada y utilizada en la aplicación, esta es *Parser.java*, la clase que define el parseador.

3. Una vez creadas las clases pertinentes es necesario compilarlas.

4. Ahora solo falta crear un fichero de texto en el que introducimos una instancia del reloj (por ejemplo, escribimos: 15:26:37), guardamos el archivo y ejecutamos el parseador:
   java Parser -file clock.txt
   Si la instancia es correcta, la ejecución de esta sentencia no produce resultado alguno; por el contrario, si la lectura está incompleta o incluye caracteres inválidos, ejecutar el parseador producirá algún tipo de respuesta.

Una vez dada la pincelada general del funcionamiento de esta herramienta, se presenta cómo está integrada en el módulo servidor. En la figura 4.9 se muestra el diagrama de flujo donde se puede ver la secuencia de operaciones que sigue el analizador para validar la regla introducida por el usuario.

Entidad de gestión Este componente es el punto central de control de las distintas unidades RPA, denominadas en otros apartados agentes de gestión. Haciendo uso de Java se han implementado las

Figura 4.9: Diagrama de flujo del analizador ABNF

funcionalidades necesarias para que desde un único equipo se pueda administrar un conjunto de RPA, en concreto, la entidad de gestión tiene una doble tarea por delante, por un lado es la encargada de traducir los datos introducidos por el usuario en objetos Java que puedan ser serializados y parseados a un XML donde se almacenan; por otro lado, su segundo cometido es enviar el fichero XML al RPA.

JAXB y sus librerías asociadas, permiten a partir de un esquema XML generar las clases Java que hacen factible crear los objetos referentes a flujos y sensores. Para ello se hace uso del comando de compilación *xjc*. Una vez generadas las clases que representan el

esquema XML, es posible convertir objetos Java desde y a un fichero XML.

El esquema XML utilizado en el presente proyecto se incluye en el anexo A de la memoria, donde se añade una breve descripción de los elementos del mismo.

Mediante las librerías de JSch para Java, en el momento en el cual el usuario selecciona el RPA al que se quiere conectar e introduce sus credenciales, se abre una conexión SSH con el RPA seleccionado; una vez abierta la conexión, el usuario modifica la configuración de flujos y sensores y envía estos cambios alojados en un archivo XML a través de SFTP, de manera que el RPA tenga una copia local de dicho XML. Cada modificación supone un proceso distinto e independiente del resto, es decir, la acción de añadir, borrar o modificar un flujo o un sensor hace que la entidad gestora acceda al XML y añada el cambio nuevo; posteriormente, se abre un canal SFTP y se envía el documento XML.

Cuando el fichero ha sido enviado, se cierra el canal de transmisión hasta que haya un nuevo cambio en la configuración. Por su parte, la conexión SSH queda abierta hasta que el usuario decide salir de la sesión a través de la interfaz.

Servidor Web Otro de los elementos fundamentales de este módulo pasarela es el servidor Web. Implementado con clases Java incluidas en la API de Servlets, las peticiones HTTP (ó HTTPS) son recibidas y procesadas mediante un Servlet denominado Controlador que se encarga de tareas como el procesamiento de la información suministrada por el usuario en los formularios.

Asimismo, este componente es el responsable del control de sesiones, es decir, es capaz de identificar si varias peticiones pertenecen a un mismo cliente. La importancia de esta capacidad es manifiesta, puesto que HTTP es un protocolo que carece de estado y para realizar un control de sesiones es necesario que cliente y servidor Web se intercambien los datos de sesión en cada petición. Esto se logra a través del objeto HttpSession comentado anteriormente.

```
HttpSession session = request.getSession(true);
```

Figura 4.10: Objeto Java con los datos de la sesión

Entre las peticiones realizadas por los clientes existen una serie de tareas comunes que son gestionadas por el contenedor de Servlets Apache Tomcat [26]. Esta no es la única herramienta externa necesaria para el correcto funcionamiento del Servlet, ya que éste debe

ser instalado en el servidor para poder redirigir las peticiones recibidas desde el cliente. La instalación o despliegue se realiza a través de un fichero conocido como descriptor de despliegue (web.xml) donde se configura el contenedor.

4.1.3. Módulo RPA

El último de los módulos que integran la plataforma consiste en una aplicación independiente del resto de elementos del sistema. Este agente de gestión está destinado a administrar la información procedente de los dispositivos y sensores instalados en el RPA de forma que habrá una réplica de él en cada avión no tripulado y cada uno funcionará de manera autónoma.

La comunicación de este módulo con el Servidor se realiza, como ya se ha explicado en secciones anteriores, a través del protocolo SFTP, que permite la transferencia del fichero XML de configuración desde el RPA al Servidor de forma segura.

Cuando un cambio es efectuado en este fichero de configuración, desde la lógica de negocio del módulo Servidor se ejecuta un script que lanza la aplicación alojada en el RPA y se comienza a leer información de los sensores y a crear flujos de datos de acuerdo a la configuración del usuario. En el momento que la aplicación decide enviar los datos recogidos, la Entidad de reenvío entra en funcionamiento creando las conexiones UDP correspondientes para que los destinatarios reciban la información.

Tecnologías software utilizadas

Las herramientas de las que se sirve este módulo para realizar sus tareas ya han sido definidas en la sección del Servidor. A excepción de HTTP que no se utiliza en este módulo, el resto de tecnologías (ABNF, SFTP, Java) se aplican de manera similar a como han sido integradas en el módulo pasarela.

Además de las anteriores, se hace uso de una librería particular de Java aplicada a las comunicaciones por puerto serie conocida como *Java RXTX*. La instalación de los controladores de esta librería hace posible que la aplicación establezca comunicaciones con los sensores integrados en el RPA a través de puerto serie.

El envío de la información leía por los sensores a los equipos que desean recibirla se realiza mediante el protocolo de transporte **UDP**. Aunque es no fiable, es un protocolo ligero y rápido, lo cual se traduce en mayor facilidad a la hora de traducir los datos de los paquetes en el equipo receptor.

Componentes del módulo RPA

Cada agente de gestión se compone de tres elementos fundamentales: Sensores, Flujos y Entidad de reenvío. Los dos primeros son los componentes que estructuran el sistema de la aplicación del RPA; el tercero, implementa la lógica necesaria para el envío de los flujos con las lecturas de los sensores.

Sensores La clase Java *Sensor* es instanciada cada vez que se integra físicamente un nuevo sensor y el usuario lo añade a través de la interfaz.

Cuando la instancia es creada se comienza a recibir lecturas por el puerto físico al que está conectado el sensor; los datos recibidos son comprobados constantemente para validar si cumplen con la regla de sintaxis que ha introducido el usuario. Hasta que no se hace efectiva la validación, cada lectura nueva es concatenada con la anterior y almacenada temporalmente. En el momento en el cual la lectura cumple con la sintaxis ABNF, se guarda junto con la marca de tiempo en la que han validado los datos y se borra del almacenamiento temporal, para que el proceso comience de nuevo con otra lectura.

Con objetivo de evitar posibles fallos y para mejorar la gestión de los *Sensores*, cada uno de ellos guarda un registro de los objetos *Flujo* que leen información de éste. En el siguiente apartado se detalla en qué consiste la clase Java *Flujo*.

Además de la regla ABNF y el registro de flujos, cada *Sensor* almacena la ubicación física del puerto al que está asociado, el nombre asignado por el usuario y los parámetros de configuración del puerto.

Flujos Un objeto Java *Flujo* es sencillamente un flujo de datos que además de las lecturas de los sensores en los que está registrado, contiene información acerca del *host* receptor (dirección IP y puerto) de los datos así como el protocolo de transporte que se utiliza para el envío de dichos datos.

Otra de las características principales de esta clase es la comprobación de la marca de tiempo de los datos que van a ser enviados. Antes de que se encapsulen en el datagrama UDP, la aplicación verifica a partir de una fecha límite introducida por el usuario al crear el *Flujo* si la marca de tiempo de los datos es demasiado antigua como para ser considerada obsoleta. De ser así, los datos a enviar son descartados y se espera a que lecturas con una marca de tiempo más actual sean recogidas.

Entidad de reenvío Este componente tiene una funcionalidad muy concreta dentro del objeto de uso de la aplicación, ya que está diseñado únicamente para que una vez descartado el estado obsoleto del flujo de datos, éste sea enviado al receptor.

Para ello, crea un socket UDP a la máquina de destino así como el datagrama donde se incluye la información de los sensores, se especifica la dirección IP del *host* y el puerto de recepción y lo envía a través del socket.

4.2. Funcionalidades de la aplicación: casos de uso y procedimiento

En esta sección se detalla el flujo de procesos para cada una de las funcionalidades que ofrece el sistema. En aras de ofrecer una guía simplificada de cada una de ellas, se ha hecho uso de distintas imágenes con capturas de pantalla de la interfaz original de la aplicación.

4.2.1. Gestión de acceso

Elección de idioma

Cuando el usuario accede mediante su navegador Web a la plataforma donde está implementado el sistema, el cliente Web le muestra la ventana que se puede ver en la figura 4.11.

Esta es la pantalla de inicio, en la que aparece en la parte superior el nombre del servicio y en la zona central dos banderas, cada una representando un idioma distinto mediante el cual se puede navegar por la aplicación.

Actualmente, está implementada una versión bilingüe de la plataforma en las dos lenguas que se han considerado más oportunas, esto es, el español y el inglés por ser la lengua franca.

Para acceder a la pantalla de acceso a la aplicación simplemente es necesario seleccionar con un *click* una de las banderas.

Procedimientos para la gestión del idioma

Para garantizar la internacionalización de la aplicación se hace uso de ficheros de propiedades llamados *bundles*. Organizados en filas, se basan en una estructura **clave = mensaje** donde la clave es común para todos los ficheros *bundle* y el mensaje varía con el idioma.

Figura 4.11: Pantalla de inicio de la aplicación

Aplicando este sistema, no es necesario crear distintas páginas web para soportar varios idiomas, sino que a través de la información de localización regional del usuario se puede reutilizar un mismo ejecutable de la aplicación.

La clase Java *java.util.ResourceBundle* permite llevar a cabo esta implementación a través del método *getLocale()* tal y como se muestra a continuación:

```
Locale locale = request.getLocale();
String languaje = locale.getLanguage();
ResourceBundle bundle = ResourceBundle.getBundle("labels", new Locale(languaje));
```

Figura 4.12: Internacionalización mediante bundles

Con el objetivo de que la aplicación pueda ser manejada por un mayor número de personas, se ha decidido establecer el inglés como idioma por defecto.

Acceso y validación del formulario

Una vez ha sido elegido el idioma, se muestra en la pantalla un formulario de acceso a la aplicación de gestión de sensores y flujos. En este formulario el usuario elige mediante un desplegable a qué RPA desea conectarse y el protocolo de acceso a estas máquinas remotas. Este protocolo de acceso es mediante el cual se produce la transferencia de ficheros entre el RPA y el servidor.

En las siguientes figuras se puede ver este formulario en sus dos versiones, español e inglés.

Figura 4.13: Formulario de acceso en español

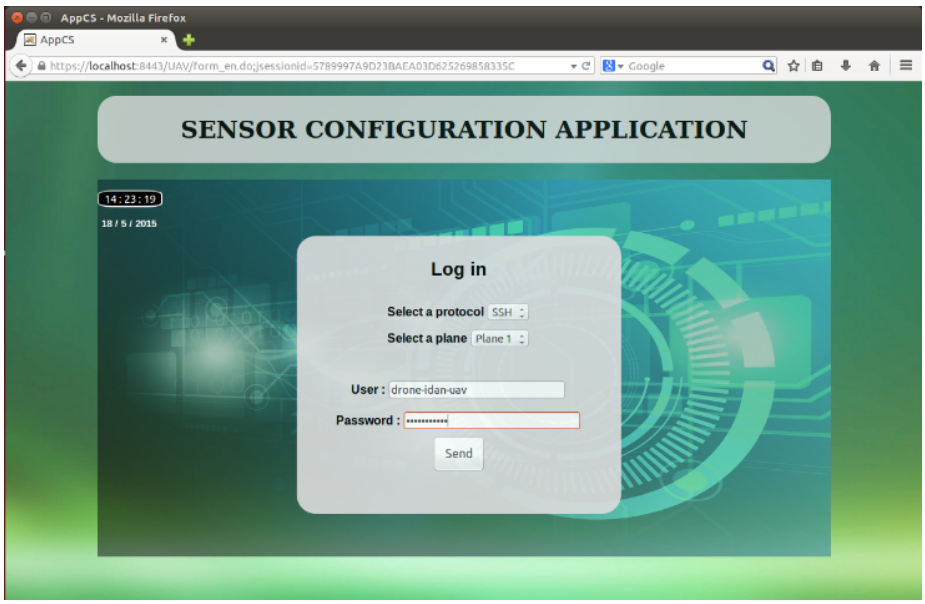

Figura 4.14: Formulario de acceso en inglés

Para poder validar el acceso es necesario conocer el usuario y contraseña correcto. En la sección de pruebas se presenta el caso

en el cual el usuario introduce las credenciales de acceso de manera equivocada y cómo es la respuesta de la aplicación ante tal supuesto.

Procedimiento para la validación del acceso al usuario

Una vez que el usuario ha introducido los datos en el formulario y ha pinchado en el botón de enviar, el controlador comprueba si es la primera vez que se accede a la aplicación para inicializar las variables pertinentes. Una vez hecho esto, es hora de validar las credenciales, para ello se comprueba si éstas coinciden con el par usuario-contraseña almacenado en el servidor; si no coinciden se lanza un mensaje de error para que el usuario vuelva a validarse, si coinciden, se crea la sesión SSH entre el servidor y el RPA seleccionado para que el usuario pueda administrar la configuración de sus flujos y sensores.

4.2.2. Gestión de sensores

Antes de mostrar la pantalla en la cual se gestionarán los sensores, es necesario realizar una serie de procesos:

- Se carga un listado de los puertos serie físicos disponibles en el RPA.
- Desde el servidor se realiza una petición SFTP del archivo de configuración XML local del RPA.
- Si no existe el archivo, se crea en el servidor; si existe, se genera un *array* de objetos Java que representan a los elementos guardados en el fichero XML de configuración.
- Los objetos son listados y mostrados al usuario a través de la interfaz, tal y como se ve en la figura 4.16.

Una vez en la pantalla de configuración de sensores, la interfaz ofrece funcionalidades distintas que se detallan a continuación.

Integración de un nuevo sensor

Previamente a añadir un nuevo sensor al RPA, el usuario debe integrarlo físicamente y a continuación instalarlo a través de la aplicación. Para ello debe rellenar todos los campos de configuración requeridos y hacer *click* en el botón de añadir (figura 4.15).

Las propiedades de configuración que el usuario debe definir son las siguientes:

Figura 4.15: Conjunto de botones de administración de la aplicación

- **ID**: número de identificación del sensor. Es un valor único generado automáticamente por la aplicación. No puede ser editador por el usuario.
- **Localización**: puerto serie físico donde está instalado el sensor. Aparece un desplegable con los puertos series físicos disponibles en el RPA.
- **Nombre**: denominación de uso coloquial del sensor.
- **Tipo de información**: sintaxis ABNF que deben seguir las lecturas del sensor.
- **Parámetros**: opciones de configuración del hardware del puerto serie al que está integrado el sensor. La implementación de este campo hace posible establecer la configuración por defecto de un puerto serie, es decir, 9600bps, 8 bits de datos, sin paridad y con 1 solo bit de parada.

Esta línea de configuración del nuevo sensor aparece debajo del listado de los sensores ya configurados.

Procedimiento para la integración del nuevo sensor

Tan pronto como el usuario selecciona el botón de añadir, el controlador recibe los parámetros introducidos desde la interfaz y asocia al nuevo sensor un identificador único. A continuación, comprueba si la regla sintáctica ABNF introducida está bien formada.

Una vez hecho esto, se añade el nuevo objeto Java *Sensor* cuyos atributos son los introducidos por el usuario en el *array* de objetos *Sensor* ya existentes (en el caso de que los haya) y se serializa el *array* para convertirlo en elementos XML que son almacenados en el fichero XML de configuración.

Figura 4.16: Pantalla de configuración de sensores

Después, el servidor envía por SFTP éste fichero al RPA con el nuevo sensor y se lanza la aplicación implementada en el RPA para que el sensor comience a capturar datos siguiendo la configuración especificada en su fichero nuevo XML.

Por otro lado, el *array* de objetos *Sensor* es listado y enviado de nuevo a la página web, donde se muestran todos los sensores integrados actualmente en el RPA.

Todo este proceso se resume en el diagrama de flujo de la figura 4.17, donde se puede ver con mayor claridad el modo de funcionamiento de los distintos elementos que hacen factible la capacidad de añadir un nuevo sensor.

Eliminación de un sensor existente

Dejar de leer datos procedentes de un sensor es tan sencillo como eliminar su configuración en la aplicación y de esta manera, borrar toda constancia de dicho elemento. A efectos de una eliminación completa, se da por supuesto el hecho de que el sensor sea desinstalado físicamente del puerto serie del RPA.

En la parte derecha de la pantalla de gestión de sensores aparece un botón rojo con un aspa en blanco por cada uno de los sensores configurados, tal y como se muestra en la figura 4.15. Pinchando en este botón se produce el borrado del sensor.

Es importante destacar que **un sensor no puede ser elimina-**

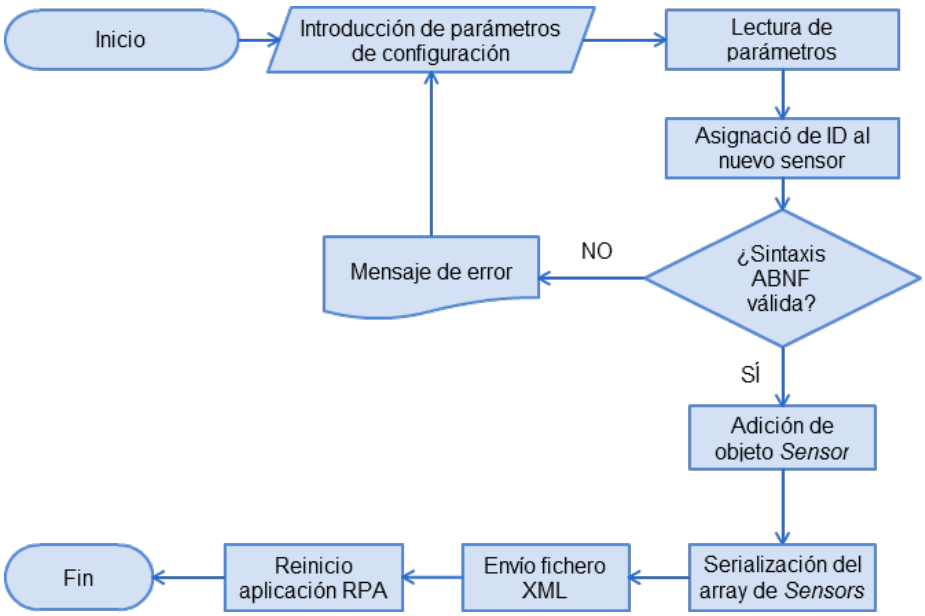

Figura 4.17: Diagrama de flujo de la inserción de un sensor

do si está asociado a un flujo de datos, es decir, si existe un objeto *Flujo* registrado en el sensor que se desea borrar, la aplicación lanzará un mensaje de error. Este supuesto se muestra en el capítulo de Pruebas y ampliaciones de implementación.

Procedimiento para la eliminación de un sensor

Cuando el usuario hace *click* en el botón de eliminar sobre un sensor al que no está asociado ningún flujo, es decir, susceptible de ser borrado, el controlador del servidor recibe la solicitud HTTP y obtiene el identificador del sensor a eliminar. Localiza en el *array* de objetos *Sensor* el que el usuario desea borrar y comprueba si está asociado a algún flujo de datos configurado.

Si no está siendo utilizado por ningún flujo, en primer lugar se borra la regla de sintaxis ABNF del fichero que registra todas las reglas ABNF de los sensores instalados y se parsea el fichero generando así las clases Java necesarias para comprobar las lecturas realizadas por los sensores, esto es, se vuelven a validar las reglas ABNF asociadas a los sensores aún instalados con el objetivo de crear únicamente las clases Javas necesarias y desechando aquellas vinculadas con la regla eliminada.

El siguiente paso es borrar el objeto *Sensor* del *array* a través de su identificador único. Una vez hecho esto, se serializa el *array* de objetos Java para crear el fichero XML con la configuración actual,

el cual es enviado por el servidor al RPA a través de SFTP. Para hacer efectivos los cambios, la aplicación del RPA es ejecutada de nuevo.

Al igual que en el caso de inserción de un sensor, una vez se ha hecho efectiva la eliminación, se vuelven a listar los sensores disponibles en la pantalla de configuración para que el usuario pueda continuar con las modificaciones pertinentes.

A continuación se incluye un diagrama de flujo con el proceso anteriormente explicado:

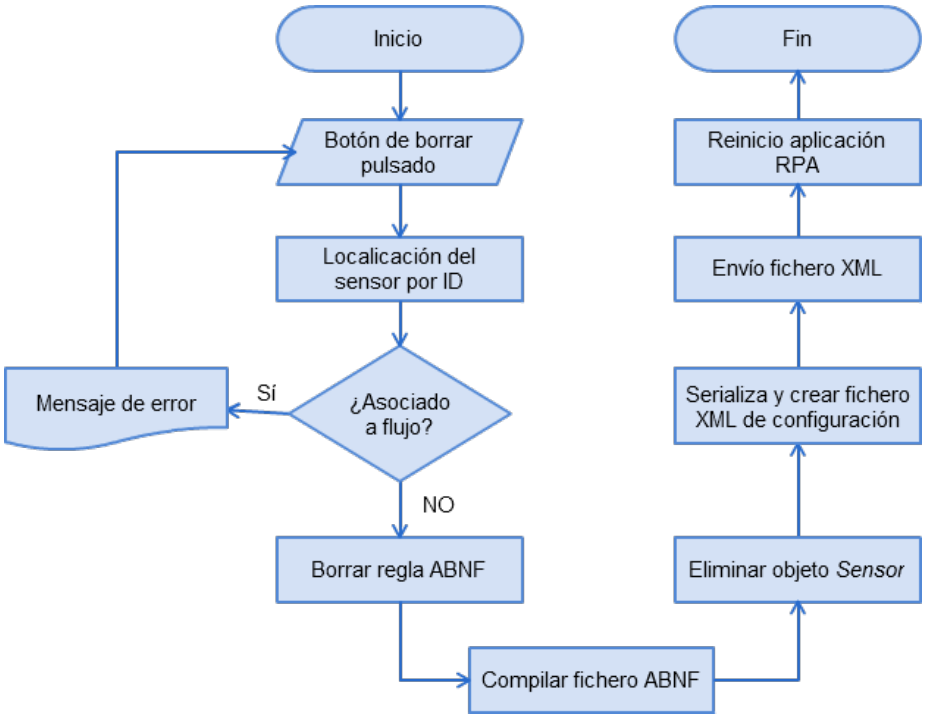

Figura 4.18: Diagrama de flujo de la eliminación de un sensor

Modificación de un sensor existente

A fin de que la configuración de los sensores a través de la plataforma Web sea lo más flexible posible, el procesado de la edición de cada campo de configuración del sensor se realiza por separado, manteniendo el resto de parámetros con los valores antiguos.

Con vista a evitar modificaciones erróneas o accidentales por parte del usuario, la capacidad de edición de éstos campos no está disponible por defecto. Para habilitarla, el usuario debe seleccionar el **botón de editar**, situado en la esquina superior izquierda de la

pantalla de configuración de sensores, donde aparece la imagen de una hoja en blanco y un lápiz (figura 4.15).

Una vez pulsado dicho botón, el conjunto de campos de todos los sensores pueden ser editados, a excepción del identificador único, que permanecerá inhabilitado.

La modificación de los sensores se realiza uno a uno y se hace efectiva una vez pulsado el **botón de guardar** (figura 4.15), situado a la derecha de los campos de configuración del sensor.

Con el fin de independizar cada funcionalidad, durante este proceso de modificación, no es posible añadir un nuevo sensor, por lo que los campos de configuración de la línea en la que se añadiría un nuevo sensor, quedan deshabilitados.

Para volver al estado original, en el que la opción de añadir está disponible y los sensores instalados son mostrados por pantalla pero sus campos no son modificables, el usuario debe hacer *click* en el **botón de retorno** (figura 4.15), situado en la parte superior izquierda del listado de sensores.

Procedimiento para la modificación de un sensor

Cuando el usuario pulsa el botón de modificar, los campos de configuración de los sensores disponibles son habilitados a la edición. Una vez se ha realizado el cambio pertinente y el usuario pulsa el botón de guardar, el controlador del servidor recibe la solicitud para hacer efectivo dicho cambio.

El procedimiento que sigue la Entidad de gestión es la siguiente: en primer lugar, se busca a través de identificador único del sensor modificado dentro del *array* de objetos *Sensor*. Después de localizar el objeto, se comprueba si el usuario ha cambiado la regla de sintaxis ABNF (campo **Tipo de información**), si ha sido modificada y está correctamente definida se continuará con el proceso, si por el contrario el usuario ha introducido la regla erróneamente, se mostrará un mensaje de error para que vuelva a escribir este campo de configuración. Una vez realizada esta comprobación, se editan los campos del sensor y se serializa el *array* para crear el fichero XML de configuración con las nuevas actualizaciones. Este fichero es enviado al RPA cuya aplicación de lectura de datos es ejecutada una vez recibido el archivo XML en cuestión.

El diagrama de flujo del procedimiento completo es el siguiente:

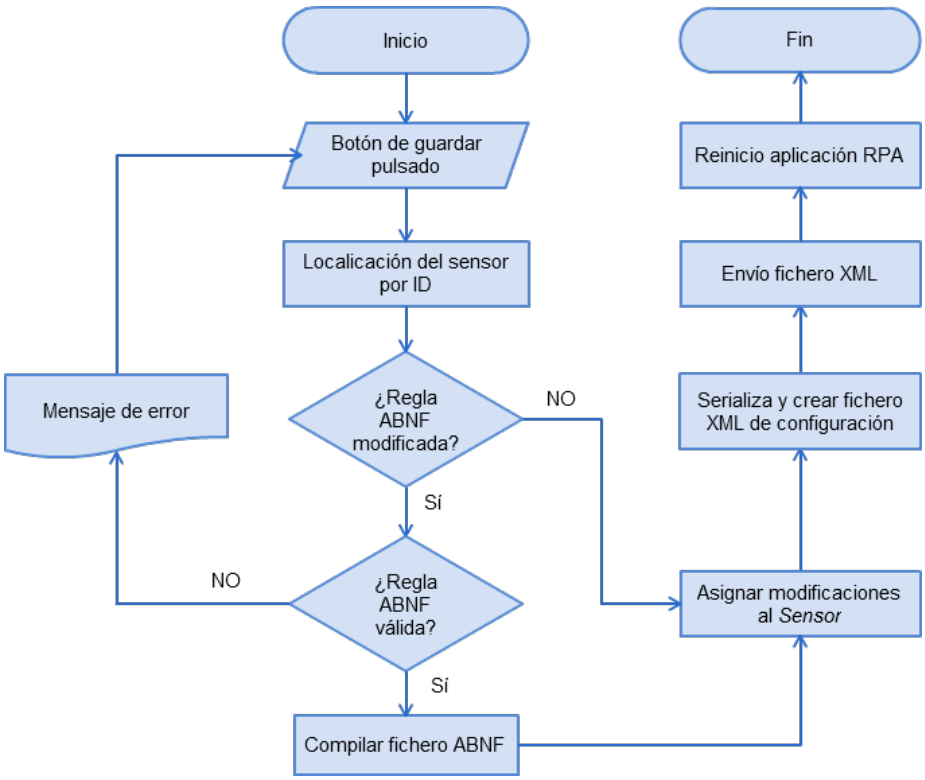

Figura 4.19: Diagrama de flujo de la modificación de un sensor

4.2.3. Gestión de flujos

La gestión de flujos se realiza cuando hay al menos un sensor instalado y configurado. Para poder administrar los flujos, internamente suceden una serie de procesos totalmente transparentes al usuario, estos son:

- Se realiza una petición SFTP del archivo de configuración XML desde el servidor al RPA.
- Se comprueba si hay algún flujo ya configurado; si existe, se muestra a través de la interfaz tal y como se puede ver en la figura 4.20.

Al igual que ocurre con la gestión de sensores, existen diversas funcionalidades asociadas a la administración de flujos. En esta sección se describe cómo el usuario puede añadir, eliminar y modificar un flujo de datos asociado a uno o varios sensores.

Integración de un nuevo flujo

Añadir un flujo nuevo a través de la interfaz Web supone que la aplicación que se está ejecutando en el RPA comience a enviar datos a una máquina destinataria (este proceso se explicará y visualizará en secciones posteriores).

Para hacer efectiva la integración de un flujo de datos nuevo, el usuario debe completar los campos de configuración requeridos, pudiendo dejar incompletos aquellos que no lo son. A continuación se presenta un listado donde se describe brevemente cada uno de ellos:

- **ID**: número de identificación único del flujo. De la misma naturaleza que su homólogo en los sensores, es asignado automáticamente por la aplicación y no puede ser modificado.

Figura 4.20: Pantalla de gestión de flujos

- **Sensores**: listado con los sensores disponibles actualmente en el RPA. Son aquellos dispositivos instalados y configurados por el usuario a través de la interfaz. Es posible elegir el orden en el que se desea recibir la información de los sensores seleccionados tal y como se muestra en la figura 4.20.
- **Periodo**: en este campo se establece el intervalo de tiempo que trascurre entre cada envío de datos a receptor.

- **Fecha límite**: indica la fecha de vencimiento de la validez de las lecturas realizadas por los sensores. Gracias a la existencia de este campo, se evita enviar al destino datos obsoletos.
- **Delimitador**: conjunto de símbolos añadidos al final de la lectura de cada sensor para identificar la información recogida por cada dispositivo.
- **Descripción**: breve reseña de las características del flujo configurado. Éste es el único campo de configuración cuya especificación no es requerida.
- **Protocolo**: establece el protocolo a nivel de transporte para el envío de los datos desde el RPA al receptor.
- **Destino**: define el nombre o dirección IP del *host* receptor así como el número del puerto donde se esperan recibir los datos.

Una vez completados los campos, el modo de proceder es equivalente al realizado para añadir un nuevo sensor, esto es, el usuario únicamente debe pulsar el botón de la figura 4.15 y la aplicación internamente gestiona la información para mostrar por pantalla el flujo configurado.

Procedimiento para la integración del nuevo flujo

Cuando el botón de añadir ha sido pulsado, el servidor recibe los parámetros configurados por el usuario y realiza dos comprobaciones: en primer lugar, verifica que el **Periodo** y la **Fecha límite** sean valores numéricos, y en segundo lugar, comprueba que el usuario haya seleccionado algún sensor del cual recibir datos. Si alguna de estas dos revisiones descubren un error, el proceso de integración no continúa y se advierte al usuario de la posible errata en dichos campos a través de la interfaz Web. Tan pronto como el servidor acepta todos los datos introducidos como válidos, el flujo nuevo es añadido al *array* de objetos flujo. A continuación el *array* es serializado y convertido a elementos XML que son almacenados en el fichero XML de configuración. Por último, el fichero es enviado por SFTP al RPA y la aplicación de recogida de datos instalada en ésta unidad es ejecutada.

Este conjunto de procesos se representa en el siguiente diagrama de flujo:

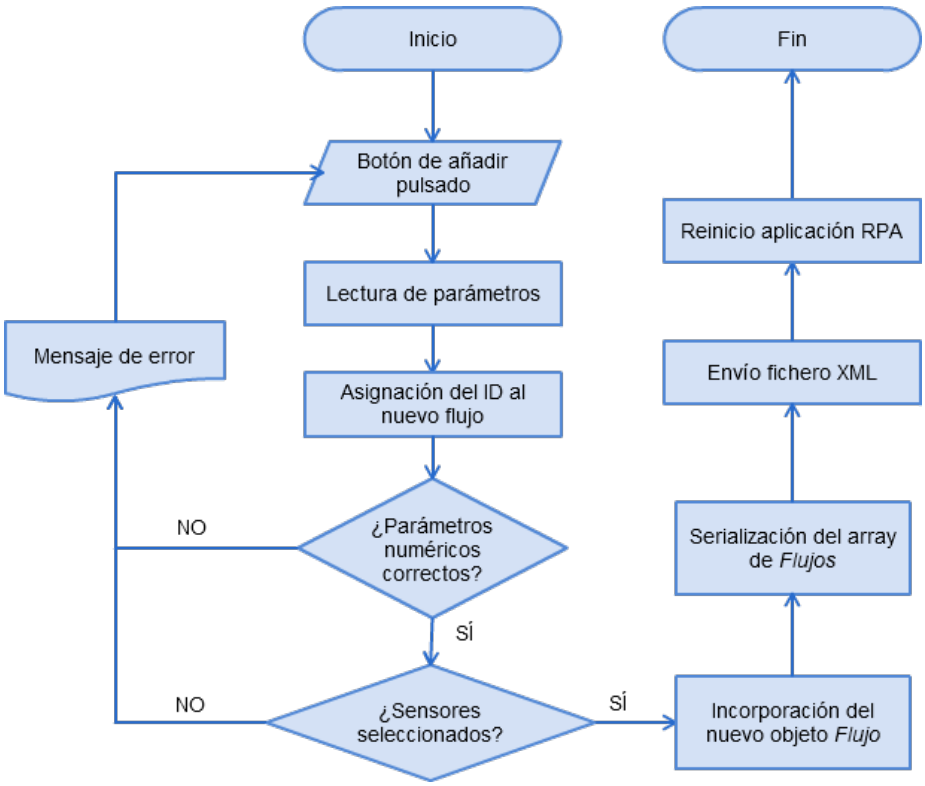

Figura 4.21: Diagrama de la integración de un nuevo flujo

Eliminación de un flujo existente

En caso de que el usuario decida eliminar un flujo ya existente, al igual que ocurre en el caso de desechar un sensor, debe pulsar el botón de eliminar (figura 4.15) que aparece en la misma horizontal que los campos de configuración de dicho flujo.

Borrar un flujo implica que la máquina destinataria de las lecturas recogidas por los sensores en los que el flujo está registrado, deja de recibir dichos datos.

Procedimiento para la eliminación de un flujo

Cuando el usuario hace *click* en el botón de eliminar sobre un flujo, el controlador del servidor Web recibe esta información y la procesa; la Entidad de gestión se encarga de acceder al número de identificación único del flujo para localizarlo dentro del *array* de *Flujos* y borrarlo.

Después, se serializa el *array* de objetos Java *Flujo* resultante para crear el fichero XML de configuración, el cual es enviado al

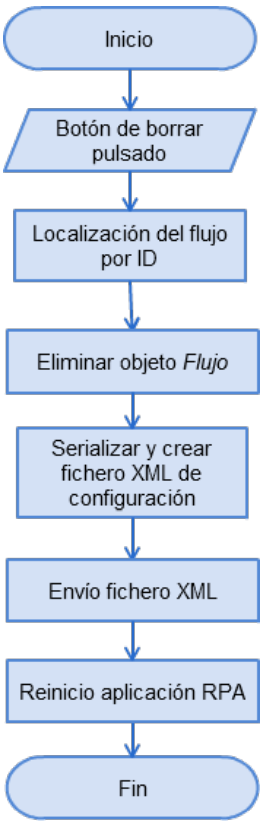

Figura 4.22: Diagrama de la eliminación de un flujo existente

RPA mediante SFTP para mantener la copia local de la unidad actualizada y se vuelve a ejecutar la aplicación de lectura de sensores instalada en ella.

Por último se muestra al usuario a través de la interfaz un listado con los flujos aún configurados en el RPA.

Para dejar claro ésta funcionalidad se presenta el diagrama correspondiente a la eliminación de un flujo de datos en la figura 4.22

Modificación de un flujo existente

La modificación de un flujo se habilita cuando el usuario pulsa el botón de editar (figura 4.15). Análogamente al procedimiento que el usuario debe seguir para modificar un sensor, al pulsar el botón antes mencionado los campos de configuración de todos los flujos, a excepción del identificador único, quedan dispuestos a ser modificados.

Del mismo modo, para guardar los cambios efectuados por el usuario, éste debe pulsar el botón de guardar (figura 4.15), o si

por el contrario se desea descartar las modificaciones, los campos de configuración quedarán deshabilitados manteniendo la configuración antigua una vez el usuario pulse el botón de retorno (figura 4.15).

Como ocurre en el caso de la edición de sensores, durante la modificación de los flujos queda descartada la opción de añadir uno nuevo. En el siguiente diagrama se presenta un esquema del procedimiento de esta funcionalidad, el cual se describe más extensamente en la próxima sección.

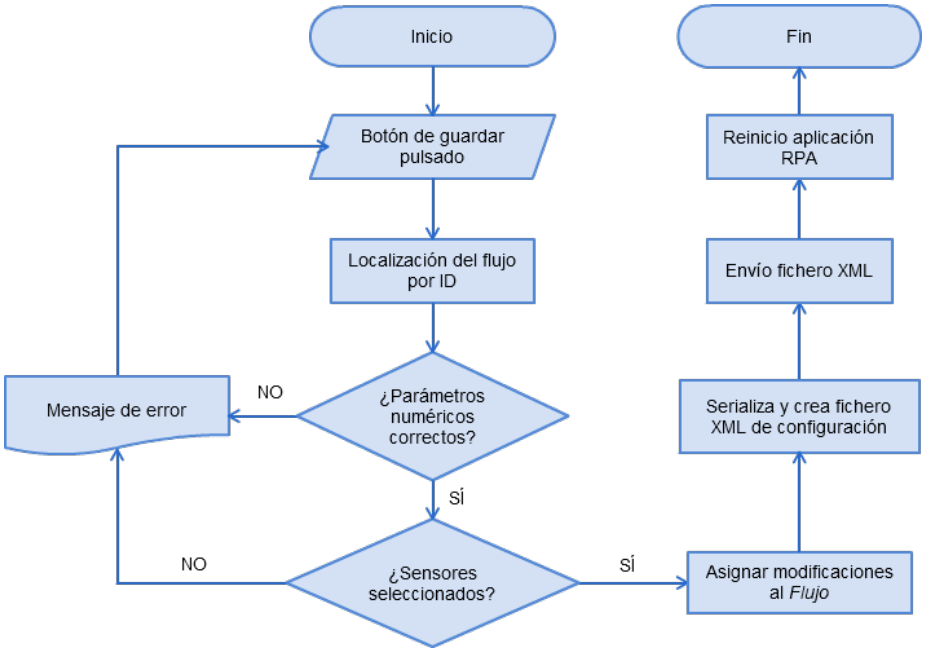

Figura 4.23: Diagrama de la modificación de un flujo

Procedimiento para la modificación de un flujo

Una vez el usuario ha decidido qué flujo desea modificar, edita los campos correspondientes y pulsa el botón de guardar, el servidor recibe una petición HTTP que le permite acceder a las modificaciones.

En primer lugar y antes de guardar los cambios efectuados en el objeto *Flujo* perteneciente al *array* de flujos, el objeto editado es localizado mediante su ID, momento en el cual se comprueba la validez de los campos numéricos y la selección de los sensores asociados, de manera análoga al procedimiento de integración de un flujo nuevo. Si hay algún error en los datos modificados, se le notificará al usuario y si no, se añadirán y guardarán los cambios de forma permanente.

A continuación, se serializa el *array* y se crea el fichero XML con la configuración de los flujos actualizada, se envía el archivo al RPA cuyos flujos han sido modificados y se vuelve a ejecutar la aplicación del RPA.

4.3. Conclusiones sobre la implementación

Para poder llevar a término las funcionalidades requeridas en el diseño, las aplicaciones desarrolladas en el RPA y servidor cuentan con cinco y cuatro clases Java respectivamente. El conjunto de estas clases han supuesto la generación de más de mil cuatrocientas líneas de código, recogidas en los métodos pertenecientes a cada clase, además de media docena de ficheros de diseño web entre JSP y CSS.

Los diagramas de clase de cada aplicación pueden verse en el anexo A, donde asimismo se incluye un listado en el cual se explica brevemente en qué consiste cada una de ellas.

En cuanto a las tecnologías ligadas al diseño web, el uso de JQuery (Javascript) para añadir capacidad de interactuación entre el usuario y la interfaz ha resultado bastante complicado; en el futuro podría plantearse la opción de implementar dichas capacidades a través de CSS3 y HTML5, ya que estas tecnologías están en auge e incorporan novedades que pueden suplir a JQuery.

Capítulo 5

Validación funcional

En este capítulo se describe cómo han sido las pruebas realizadas para verificar el correcto funcionamiento de las aplicaciones desarrolladas, así como el resultado obtenido.

5.1. Pruebas de usabilidad

La funcionalidad que ofrece el sistema se ha comprobado a través de una evaluación de usabilidad de la plataforma web, es decir, se ha revisado la interfaz de forma exhaustiva llegando a las siguientes conclusiones:

- El usuario puede elegir el idioma en el que desea navegar por la interfaz, además, puede volver en todo momento a la pantalla de inicio para cambiar el idioma.
- Los dos elementos principales configurables son fácilmente reconocibles y el usuario puede alternar entre ambas configuraciones sin problemas.
- Aparecen mensajes de error cuando el usuario introduce datos erróneamente junto con una breve explicación sobre lo que la aplicación espera recibir.
- El usuario puede comprobar en todo momento que está registrado en la aplicación, ya que su nombre aparece en todas las vistas de la página web durante la sesión.
- El usuario es capaz de añadir, borrar o modificar cualquier flujo o sensor, siempre que cumpla con las condiciones necesarias en el caso de borrado de flujos de datos.
- La navegabilidad a través de la interfaz es sencilla, pues la información aparece de forma concisa para no confundir al usuario.

- La accesibilidad de la plataforma web para personas con pequeños problemas de visión está soportada, ya que se usa un rotulado con letras grandes, además de un alto contraste entre el color del fondo y el de fuente.

5.2. Pruebas de concepto

En esta sección se desea ratificar el correcto funcionamiento de las aplicaciones desarrolladas, para ello se ha recurrido a una serie de pruebas de concepto donde el resultado de cada una de ellas es mostrado a través de capturas de pantalla del analizador de tráfico de red Wireshark [28].

Las pruebas realizadas se han efectuado en una maqueta propiedad del Departamento de Telemática de la Universidad Carlos III de Madrid. Aunque los componentes de este prototipo se describen en el anexo B, a continuación se incluye una ilustración que muestra la relación de los elementos que componen la maqueta (figura 5.1).

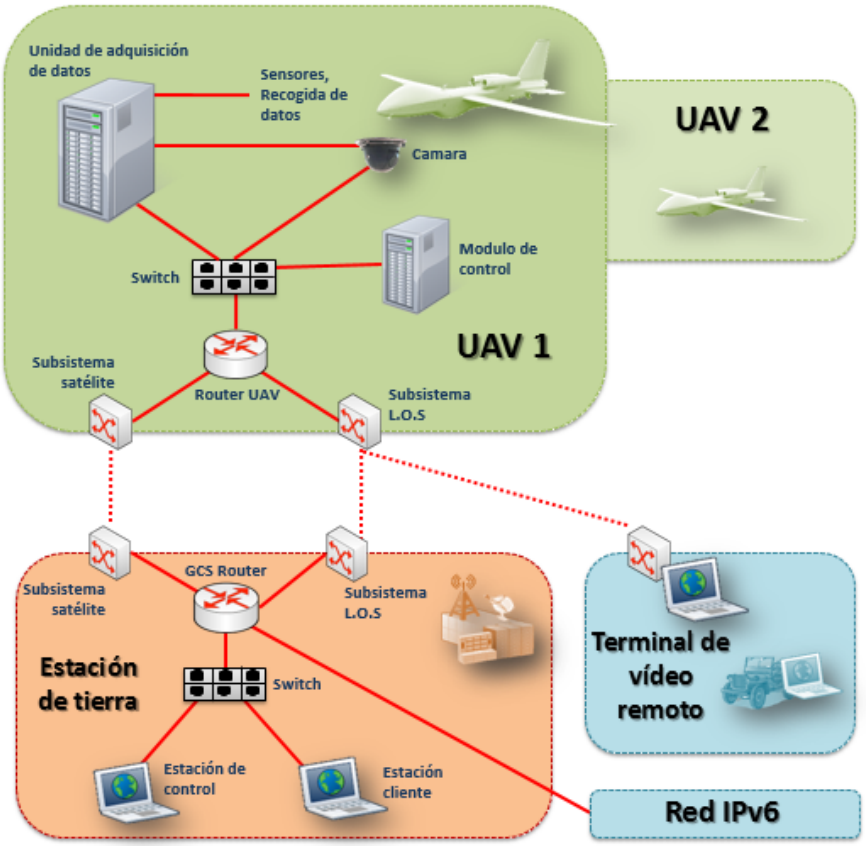

Figura 5.1: Maqueta de pruebas

Dentro de este esquema el sistema de configuración de sensores se ha desplegado en tres de los elementos que componen la maqueta, en concreto, estos son:

- **Unidad de adquisición de datos**: en ella se instala y configura la aplicación del módulo RPA pues en ella van integrados los sensores que recogen los datos del exterior.
- **Estación de control**: es utilizado como módulo servidor, es decir, en este componente se despliega el servidor web así como las librerías necesarias para que funcione como módulo pasarela entre el cliente y el RPA.
- **Estación cliente**: punto de acceso a la plataforma web desde la cual el usuario se conecta con el RPA que desea configurar.

5.2.1. Instalación y configuración de varios sensores

En la primera prueba, se utiliza un RPA sin ningún sensor ni flujo configurado previamente. Tras elegir el idioma, el usuario inicia sesión en la aplicación para acceder al RPA en cuestión. Se configuran tres sensores entre los cuales sólo uno está instalado físicamente en el RPA, este consiste en una placa Arduino Ethernet con 32KB de memoria flash y SRAM de 2KB [2] al que se le ha acoplado un sensor de temperatura LM35 DZ. El resto de sensores son creados a través de puertos lógicos en los que se vuelca información constantemente.

A continuación se configuran dos flujos, cada uno registrado en sensores diferentes. El primero de estos flujos, se asocia al sensor de temperatura, mientas que el segundo flujo está asociado a los sensores virtuales restantes además del sensor de temperatura. La periodicidad de los flujos creados es de uno y dos segundos y medio, respectivamente.

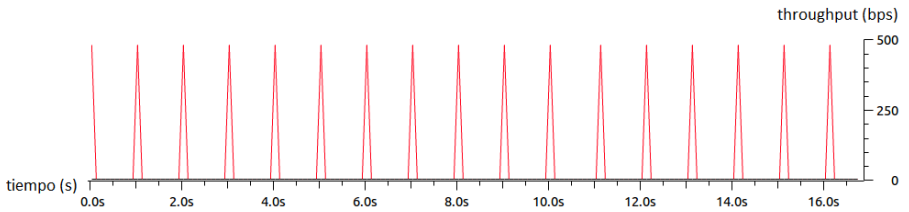

Figura 5.2: Gráfica resultante de añadir un flujo

En la gráfica de la figura 5.2 se puede ver cómo varía el flujo de paquetes UDP que llegan al puerto del *host* receptor. Inicialmente, solo se recibe información de uno de los flujos; cada paquete

UDP enviado con datos del este flujo es representado mediante un triángulo rojo [1].

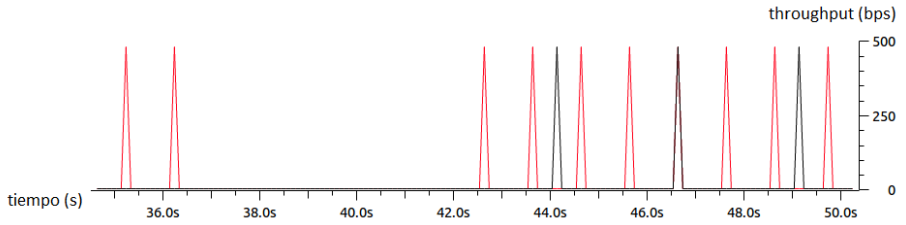

Figura 5.3: Gráfica resultante de añadir dos flujos

Tras lanzar el primer flujo, el segundo es creado. Para ello, la aplicación ha de reiniciarse, motivo por el cual entre los segundos 36 y 42 no se recibe ningún paquete UDP tal y como se indica en la figura 5.3. Una vez reiniciada la aplicación en el RPA, se comienzan a crear paquetes UDP de ambos flujos, los que se tenía inicialmente representados en rojo, y unos nuevos en color negro que son enviados cada dos segundos y medio tal y como el usuario ha configurado previamente.

Por otra parte, para evaluar el ancho de banda consumido por ambos flujos de datos, en primer lugar, se debe tener en cuenta que el tamaño mínimo del *payload* de una trama Ethernet es de 48 bytes (la trama en total son 64 bytes). Tanto en el primer flujo de datos como en el segundo flujo, este tamaño mímino no es alcanzado por lo que es necesario añadir *padding* para crear la trama. Esto hace que el tamaño de los paquetes de ambos flujos sea el mismo, y por tanto se puede observar en la gráfica que es mucho más eficiente en cuanto a consumo de ancho de banda asociar la información de varios sensores en un mismo flujo, que enviar los datos de cada sensor en flujos distintos.

5.2.2. Modificación de un elemento

Una vez evaluado el comportamiento de las aplicaciones cuando se crean varios sensores y flujos, es momento de comprobar qué ocurre al modificar un campo cualquiera de uno de los flujos de datos.

Para realizar dicha evaluación, se ha elegido **editar la periodicidad del segundo flujo** creado; a partir de ahora, los paquetes UDP serán enviados al *host* receptor **cada tres segundos**.

La figura 5.4 permite ver el cambio en el tráfico UDP que llega a los puertos de recepción de la máquina del destinatario. En el

[1] Cada pico representa un paquete por marca de tiempo.

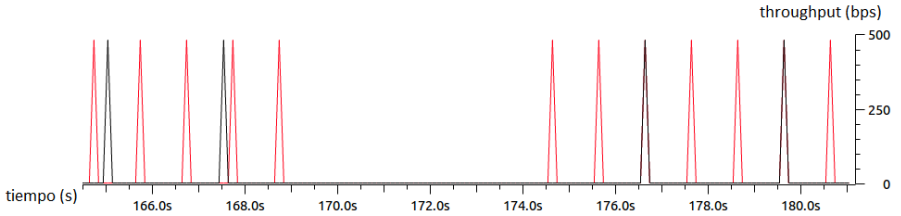

Figura 5.4: Gráfica resultante de modificar un flujo

segundo 174 las modificaciones se hacen efectivas y la periodicidad del flujo negro aumenta, solapándose ambos flujos cada 3 segundos.

5.2.3. Borrado de todos los elementos configurados

Por último se verifica qué ocurre con el tráfico de datos cuando todos los flujos configurados son eliminados de la aplicación.

El resultado esperado es que no se genere tráfico UDP alguno entre el RPA y el *host* receptor.

Tras realizar la comprobación con Wireshark, se obtienen las siguientes gráficas:

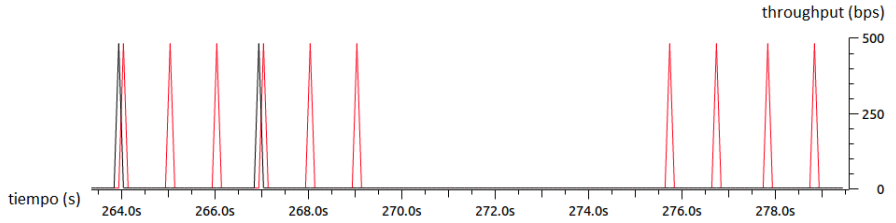

Figura 5.5: Gráfica resultante de eliminar un flujo

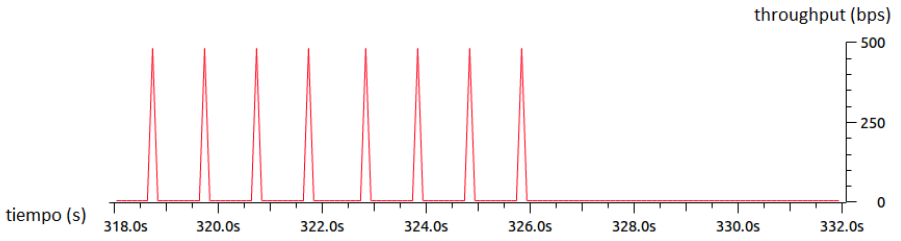

Figura 5.6: Gráfica resultante de eliminar todos los flujos

En la primera figura, se puede comprobar cómo en el segundo 267 se deja de recibir tráfico UDP por el puerto asociado al segundo

flujo creado (color negro). Esto se debe a que en ese momento la aplicación RPA hace efectiva la eliminación del flujo realizada por usuario, se reinicia y comienza a recibir tráfico de un solo flujo a partir del segundo 276.

Para terminar, la figura 5.6 muestra el comportamiento del tráfico UDP cuando el último de los flujos disponibles es eliminado. En el instante 326, el usuario borra a través de la plataforma web el flujo y se reinicia la aplicación en el RPA, la cual elimina definitivamente el flujo de datos, lo que supone dejar de recibir información por el puerto del *host* receptor asociado a dicho flujo.

Capítulo 6

Planificación del proyecto

En este capítulo se recogen los aspectos del proyecto relativos a su planificación, medios técnicos utilizados y coste económico asociado al mismo.

6.1. Modelo del ciclo de vida seleccionado

Una vez definidas las fases del proyecto y los requisitos principales se determina el modelo de ciclo de vida que se ha de seguir para finalizar el proyecto en el tiempo previsto.

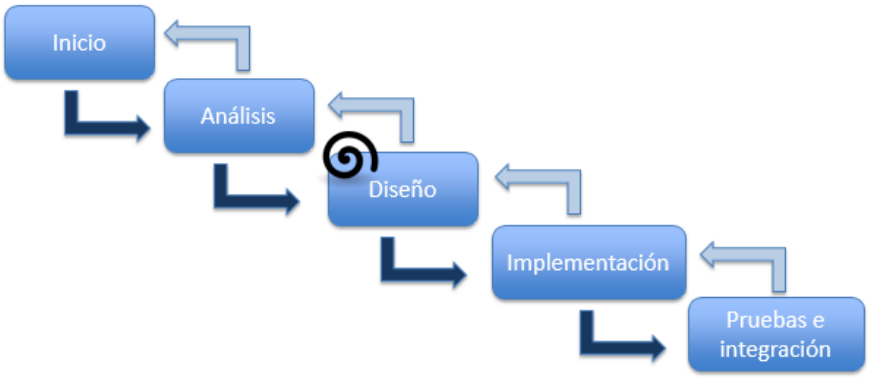

Figura 6.1: Modelo en cascada con reducción de riesgos

El tipo de ciclo de vida escogido es el de cascada modificada, en el cual se combinan los modelos tradicionales de cascada y espiral. Esto es muy útil en proyectos cuya definición no es estable como ocurre en el presente proyecto, donde el diseño está suscito a cambios progresivos que se vayan dando a lo largo del análisis de las tecnologías necesarias para el desarrollo del sistema.

Precisamente, la elección de un modelo de cascada modificada es debida al riesgo que conlleva el uso de tecnologías desconocidas. En aras de minimizar este riesgo se aplica el ciclo de vida espiral en la fase de diseño, lo que permite cambiar el enfoque sin retrasar el conjunto del proyecto.

Por último, es importante mencionar que tanto el desarrollo de la aplicación instalada en el GCS como la del RPA siguen este tipo de ciclo de vida.

6.2. Gestión del trabajo

Nombre de la tarea	Inicio	Final	Duración (días)
Inicio	**01/09/14**	**15/09/14**	**11**
Análisis del estado del arte	**16/09/14**	**30/09/14**	**11**
Análisis de tecnologías web disponibles	16/09/14	19/09/14	4
Comparación lenguajes de programación	16/09/14	18/09/14	3
Elección del entorno de desarrollo	22/09/14	24/09/14	3
Configuración herramientas externas	24/09/14	30/09/14	5
Aplicación en GCS	**02/10/14**	**24/12/14**	**60**
Análisis de los requisitos principales	02/10/14	03/10/14	2
Diseño del módulo de acceso	06/10/14	14/10/14	7
Diseño del módulo servidor	13/10/14	22/10/14	8
Diseño de la entidad de gestión	15/10/14	31/10/14	13
Diseño de las comunicaciones entre servidor y RPA	27/10/14	31/10/14	5
Codificación de los módulos de acceso y servidor	02/11/14	08/12/14	27
Pruebas de la aplicación GCS	09/12/14	24/12/14	12
Aplicación en RPA	**19/12/14**	**13/02/15**	**41**
Análisis de los requisitos	19/12/14	23/12/14	3
Diseño del módulo RPA	07/01/15	14/01/15	6
Codificación módulo RPA	12/01/15	06/02/15	20
Pruebas módulo RPA	09/02/15	13/02/15	5
Integración	**16/02/15**	**27/03/15**	**30**
Redacción de la memoria	**22/03/15**	**10/06/15**	**59**

Figura 6.2: Tabla de planificación del proyecto

La planificación del trabajo que se ha seguido durante los meses de desarrollo del proyecto se muestra en el diagrama de Gantt de

la figura 6.3. Las tareas y subtareas que componen este diagrama vienen descritas en la tabla previamente presentada 6.2.

Figura 6.3: Diagrama de Gantt

En el Gantt se puede observar que el desarrollo de muchas fases es lineal, esto se debe a que, por ejemplo en el caso de las pruebas de ambas aplicaciones, es necesario haber completado su implementación (aunque luego se realicen cambios concretos) para poder realizar la fase de pruebas.

Por el contrario, las etapas de diseño y codificación pueden solaparse en cierta medida ya que los detalles específicos del diseño no son necesarios para implementar funcionalidades básicas de la aplicación.

6.3. Medios técnicos empleados

En esta sección se presenta el conjunto de materiales tanto hardware como software que han resultado necesarios para la realización del proyecto.

6.3.1. Herramientas hardware

A nivel de materiales físicos, se ha necesitado el siguiente conjunto de medios:

- Nodo Inteligente de Adquisición de Datos (IDAN) de la empresa RTD Embedded Technologies, Inc. [59] como módulo instalado en el RPA para recoger datos de los sensores.
- El servidor está integrado en un PC de sobremesa con procesador Intel Core i5 y sistema opertivo Ubuntu 14.04 LTS.
- Portátil Dell con procesador Intel Centrino Duo y Linux Debian Squeeze como sistema operativo.

Además de estos, objetos tales como pantalla, teclado, ratón y dispositivos con conectores puerto serie han sido requeridos para la integración del sistema completo.

6.3.2. Herramientas software

El entorno de desarrollo escogido para la programación de las aplicaciones instaladas en el GCS y RPA ha sido Netbeans IDE 8.0.1 [15] junto con su extensión para Java EE y el servidor web Apache Tomcat.

Para lograr la funcionalidad requerida en el sistema, han sido utilizadas las siguientes librerías externas:

- JAXB [57] imprescindible para asignar clases Java a representaciones XML.
- RXTX [61] necesario para gestionar con Java las comunicaciones a través del puerto serie.
- JSch [36], la implementación en Java para utilizar SSH2 (ó SSH seguro).
- aPArse [55], librería gracias a la cual es posible leer gramática ABNF y producir clases Java.

En cuanto a los medios de planificación, se ha hecho uso de la herramienta online *SmartSheet* [65] para la realización del diagrama Gantt.

Por su parte, la redacción de la memoria ha sido llevada a cabo mediante el editor TeXstudio [8], un entorno de código abierto que permite crear documentos en LaTeX, y MiKTeX [62], uno de los compiladores de este sistema de composición de documentos para Windows.

6.4. Análisis económico del proyecto

En este apartado se indican los costes asociados a los materiales necesarios para el desarrollo del sistema, así como una serie de estimaciones ligadas al coste de personal que se detalla más adelante.

En la tabla que se presenta en la figura 6.4 no aparecen elementos tales como costes de licencias u otras herramientas externas; esto es debido a que el conjunto del software utilizado en este proyecto utiliza licencias públicas GNU.

Descripción	Coste	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador portátil Dell	300,00 €	100	9	60	45,00 €
PC de sobremesa Acer	500,00 €	100	9	60	75,00 €
Nodo IDAN	7.233,38 €	100	9	60	1.085,01 €
Sensores	39,78 €	100	9	60	5,97 €
Otros materiales	180,00 €	100	9	60	27,00 €
				Total	**1.237,97 €**

Figura 6.4: Tabla de amortizaciones de los materiales

La amortización de los materiales se ha obtenido mediante la siguiente fórmula:

$$\frac{A}{B} \times C \times D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

Figura 6.5: Fórmula de cálculo de las amortizaciones

En relación con los costes de personal, éstos se han calculado partiendo de la base de cuatro horas al día trabajadas durante los 388 días laborales que hay entre los meses de Septiembre de 2014 y Enero de 2015, ambos incluidos (sin contar los días de cierre de la universidad por festivos). A partir de Febrero de 2015, el tiempo dedicado a la realización del proyecto se ajusta a dos horas al día

hasta el mes de Junio (no incluido), lo que hace un total de 543 tal y como se indica en la figura 6.6.

Apellidos y nombre	Categoría	Dedicación (horas totales)	Coste por hora	Coste
Lucía Oliva Bállega	Ingeniero	543	20,50 €	11.131,50 €
Iván Vidal Fernández	Ingeniero Senior	60	33,00 €	1.980,00 €
			Total	13.111,50 €

Figura 6.6: Presupuesto final del proyecto

La amortización se ha calculado aplicando la siguiente fórmula:

En el cálculo del coste de personal también se han incluido las horas de tutorías tal y como se indica en la nueva Normativa General para trabajos fin de grado.

Por último, considerando que la tasa de costes indirectos es del 20 %, el presupuesto total queda resumido en la siguiente figura:

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	13.111,50 €
Amortización	1.237,97 €
Costes Indirectos	2.869,89 €
Total	**17.219,37 €**

Figura 6.7: Presupuesto final del proyecto

Capítulo 7

Conclusiones y líneas futuras

En este capítulo se presentan las conclusiones generales del proyecto así como los diversos retos encontrados durante su desarrollo e implementación. Asimismo, los resultados obtenidos son comentados y finalmente se introduce un conjunto de ideas para desarrollos futuros sobre el sistema desplegado.

7.1. Conclusiones sobre el proyecto

El desarrollo del presente proyecto se ha orientado a conseguir el sistema de configuración de sensores más flexible posible. Para lograrlo, varias tecnologías fueron integradas de tal modo que trabajaran de manera conjunta.

La integración de todos los componentes se sintetiza en el despliegue de dos aplicaciones divididas en tres módulos, los cuales están conectados entre sí. La idea de modularizar el sistema de esta forma, permite facilitar la detección de errores así como el desarrollo de futuras ampliaciones del proyecto.

Es importante señalar que el sistema ha sido validado utilizando un número reducido de sensores. Sin embargo, la limitación del número de sensores viene determinada por la cantidad de puertos serie físicos del RPA y no por la propia aplicación.

7.1.1. Desarrollo del proyecto

El desarrollo del proyecto ha sido llevado a cabo en módulos. En primer lugar, el módulo relacionado con el lado del cliente fue implementado debido a la necesidad de tener capacidad de interactuar con la aplicación.

A continuación, la lógica detrás del módulo servidor fue programada; en esta etapa se utilizaron lenguajes sobre los cuales no se tenía una experiencia previa como la gramática ABNF y el esquema XML. Antes de hacer disponible la conexión entre los módulos servidor y RPA, fue necesario un estudio sobre el modo de funcionamiento de ABNF y de qué manera se podía integrar esta tecnología en el proyecto.

En primera instancia, el hecho de utilizar tecnologías hasta el momento desconocidas parecía aparentemente complicado. Sin embargo, la idea inicial de dividir el desarrollo del proyecto en módulos pequeños resultó ser completamente apropiada ya que facilitó la implementación de todo el sistema así como la integración de los diferentes elementos que lo componen.

Dentro de la fase del despliegue de este módulo, uno de los mayores retos a superar fue el de las interrupciones que sucedían cuando el servidor necesitaba conectarse al RPA a través de SSH. Para garantizar la seguridad de la conexión entre ambos módulos, el servidor SSH (instalado en el RPA) tenía activa la autenticación por contraseña, con lo cual cada vez que se solicitaba acceso al RPA remoto, las credenciales de usuario eran requeridas para que este acceso fuera concedido y el programa pudiera continuar con su ejecución.

Con vistas a evitar este problema se decidió utilizar la autenticación basada en clave pública; con la introducción de este cambio, se lograba mejorar la seguridad y eliminar definitivamente los cortes de ejecución, ya que el servidor SSH no solicitaba al usuario sus credenciales para concederle acceso, pues tenía guardada la clave pública compartida por ambos.

Más tarde, cuando las validaciones funcionales fueron efectuadas, aparecieron diversos retos relacionados con los sensores, puesto que no se disponía del suficiente número de sensores físicos que pudieran ser conectados a los puertos series disponibles en el RPA para comprobar el comportamiento de la aplicación. Este problema fue solucionado mediante la creación de varios puertos lógicos, cada uno de los cuales estaba ligado a un puerto serie, en los que se escribían datos constantemente.

Finalmente, el último gran asunto abordado fue la búsqueda de un software que hiciera posible la conexión a puertos serie mediante el lenguaje Java. El primer enfoque fue utilizar la API de comunicaciones de Java; no obstante, esta carece de soporte actualmente. La alternativa propuesta fue RXTX, una API que permite sin problemas la comunicación entre los sensores físicos y la aplicación instalada en el RPA.

Una vez superado el problema, el resto del desarrollo del proyecto

fue abordable gracias a la gran cantidad de información relacionada con Java, consecuencia de que siempre exista una librería para implementa cualquier funcionalidad requerida.

7.1.2. Resultado obtenido

Después de estos meses de trabajo y a pesar de las dificultades asociadas a trabajar con tecnologías sobre las cuales no se disponía de un conocimiento previo, el objetivo de crear un sistema **flexible** capaz de leer datos con una sintaxis concreta y enviar la información recogida con una periodicidad configurable, entre otras características, ha sido conseguido.

Por su parte, las vistas de la interfaz gráfica en el lado del cliente son estéticas a la par que intuitivas. Sin embargo, es cierto que podrían incluirse algunas ventanas de ayuda para guiar al usuario a través de la interfaz, ya que por ahora únicamente aparecen mensajes de error cuando el usuario introduce datos equivocados.

El funcionamiento del resto de componentes del sistema es correcto y a pesar de que la espera entre cambios en la configuración podría ser mejorada, se considera que el resultado global obtenido es satisfactorio.

7.2. Líneas futuras

En esta sección se mencionan nuevas líneas de trabajo relacionadas con el proyecto.

7.2.1. Nuevo soporte para conexiones por puerto serie

En la actualidad, la conexión software entre los sensores y el RPA se realiza a través de una librería de Java denominada RXTX, la cual permite a la aplicación RPA leer información de los sensores instalados en este dispositivo.

Desde la versión 8 de Java, la librería previamente mencionada deja de estar soportada si no se realizan una serie de modificaciones; por esta razón, en el futuro será necesario actualizar la API utilizada para comunicar sensores y RPA.

Una de las librerías más utilizadas hoy en día para trabajar con puertos serie es la denominada jSSC (del inglés, *Java Simple Serial Connector*). A pesar de que este proyecto de código abierto es aún inestable bajo determinados usos, se caracteriza por una simpleza que hace posible la migración desde RXTX a esta API sin que surjan gran número de dificultades. Debido a ello, la idea de esta línea de

trabajo futuro es adaptar la implementación de la aplicación RPA para que la librería jSSC pueda ser integrada y en consecuencia, evitar la obsolescencia del conjunto del sistema.

7.2.2. Nuevos protocolos de transporte

El protocolo UDP proporciona una forma rápida de transmitir datos ya que el remitente no espera para asegurarse de que el destinatario ha recibido el paquete. Debido a que el sistema implementado debe transmitir rápidamente la información recogida por los sensores, este ha sido el protocolo de transporte utilizado.

Sin embargo, en el futuro el usuario puede buscar una garantía en la entrega de los datos transmitidos, requisito que el protocolo UDP utilizado hasta ahora es incapaz de cumplir ya que no proporciona control de flujo ni corrección de errores.

Una línea de trabajo potencialmente interesante a este respecto es la incorporación de nuevos mecanismos para la entrega de la información desde las unidades RPA. A modo de ejemplo, podría incorporarse el soporte de TCP, para de esta forma poder proporcionar un servicio fiable orientado a conexión. También es interesante de cara a desarrollos futuros, considerar los nuevos protocolos y mecanismos que actualmente se están desarrollando los escenarios emergentes de la Internet de las cosas (IoT- Internet-of-Things), como CoAP [64], o los nuevos mecanismos y protocolos de entrega centrados en la información (Ej.: Data-Distrubtion Service – DDS [1]).

En relación con la posibilidad de utilizar TCP, a fin de poder incorporar este protocolo de transporte, la aplicación desplegada en el RPA necesitaría ser modificada. Actualmente, el envío de información de los sensores hasta el receptor mediante paquetes UDP es llevado a cabo por una única clase Java denominada ***Forwarded***; por ello, añadir un nuevo protocolo de transporte sería posible simplemente incorporando otra clase Java la cual implementara los métodos pertinentes para poder crear y enviar los paquetes TCP.

7.2.3. Sistema de reenvío de información

En algún momento se puede dar la situación de tener dos usuarios, uno operando desde la estación de control en tierra y otro trabajando desde una red externa a dicha estación (denominada *"Centros de control externos"* en la figura 3.1 del capítulo 3). Con el sistema actual, si ambos clientes se registran en la misma unidad RPA, la

[1] Data Distribution Service Portal. http://portals.omg.org/dds/

aplicación desplegada en el dispositivo enviará dos veces la misma información.

La idea que se quiere desarrollar para evitar esta situación de datos duplicados, es la de diseñar un sistema en el módulo servidor que reenvíe la información recibida en la estación de tierra desde ella hasta todos los clientes conectados por redes externas. Haciendo uso de este procedimiento se mejoraría el rendimiento del sistema ya que la información sólo sería enviada una vez desde el RPA.

7.2.4. Optimización del tiempo

Tal y como se pudo comprobar en el capítulo 5, cuando un cambio en la configuración de cualquier sensor o flujo de datos es efectuado, la aplicación instalada en el RPA se reinicia. En la actualidad, el tiempo de reinicio son aproximadamente 6 segundos, una espera ligeramente superior a la deseada.

En consecuencia, uno de los planteamientos que pueden ser implementados en el futuro es el de optimizar este tiempo de reinicio. Para ello, se ha de tener en cuenta que este depende del tiempo de compilación y ejecución del programa instalado en el RPA. A pesar de que el comando de Java para compilar ficheros, **javac**, no tiene opciones para optimizar la compilación, es posible utilizar el intérprete de Java para ejecutar programas con opciones como *-Xmixed*, la cual consigue que el código repetitivo sea ejecutado de manera más eficiente.

Por otro lado, es posible realizar un estudio sobre otros compiladores Java para determinar si existe alguno que pueda garantizar un mejor rendimiento que el obtenido hasta el momento.

7.2.5. Soporte de seguridad

Actualmente, el cifrado de los datos enviados desde el RPA hasta los receptores interesados en la información, solo se garantiza a nivel de enlace y red a través de túneles IPsec[2]. Aunque, en primera instancia, el alcance del proyecto no engloba la tarea de ocultar el contenido de la información enviada, sí que es interesante añadir como futura línea de trabajo una herramienta que ofrezca soporte de seguridad a nivel de aplicación que garantice que únicamente los equipos de origen y destino son capaces de acceder a los datos recogidos por los sensores.

[2]IPsec está implementado en la maqueta de pruebas disponible en el Departamento de Telemática de la Universidad Carlos III de Madrid, utilizada para desplegar y validar la plataforma.

Anexos

Anexo A

Diagramas de clase y esquema XML

El presente anexo recoge los distintos diagramas de clases de las aplicaciones desarrolladas, así como el esquema XML con la estructura de los ficheros de configuración XML.

A.1. Diagramas y descripción de clases

Esta sección incluye los diagramas de clase de las dos aplicaciones que dan forma al conjunto del sistema, éstas son, la alojada en el RPA y la ubicada en el módulo servidor, así como una breve descripción de cada una de las clases.

A.1.1. Aplicación del RPA

En la figura A.1 se presentan las clases Java pertenecientes al paquete *rpa*. Además de este paquete existen otros dos, *abnf* y *beans*, los cuales se describen a continuación:

- Paquete **abnf**: incluye la clase *Parser* necesaria para comprobar si la regla ABNF introducida por el usuario en el momento de la configuración de un sensor nuevo es válida.
- Paquete **beans**: para poder acceder al fichero XML de configuración y manipular los elementos que lo componen, es necesario convertirlos a objetos Java, instancias de clases que se almacenan en este paquete.

Con vistas a sintetizar la información relacionada con las clases del paquete *rpa*, a continuación se explica de manera concisa la principal característica de cada una de ellas:

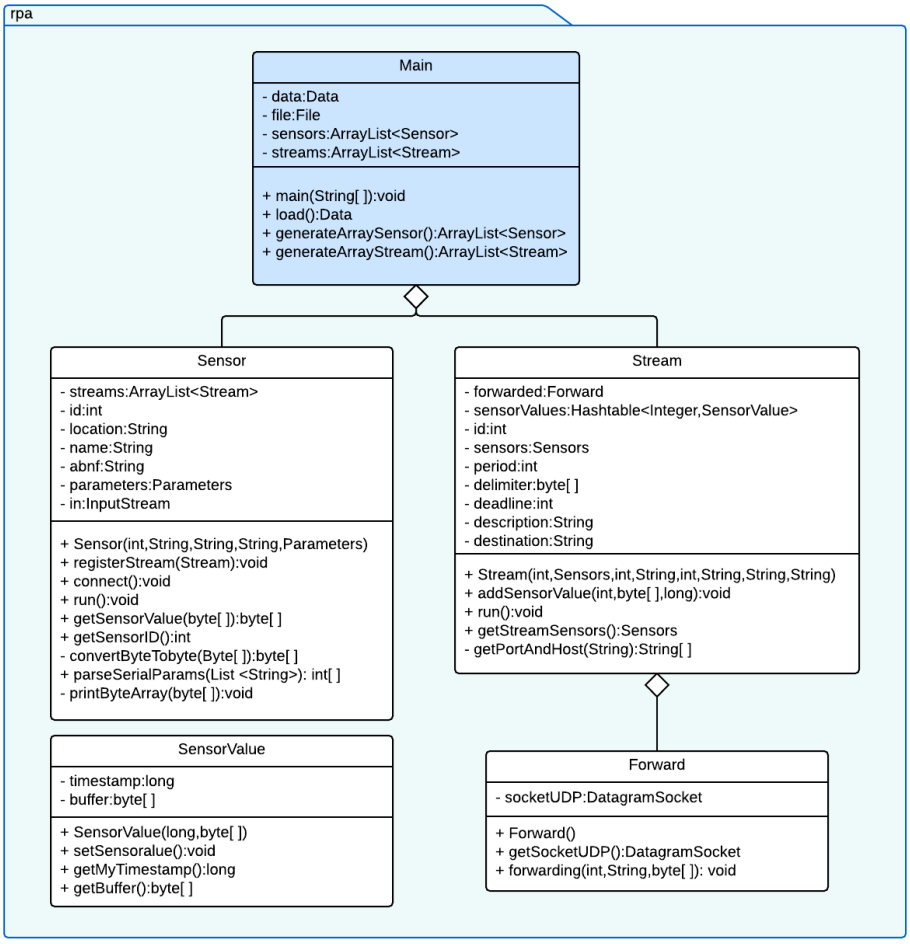

Figura A.1: Diagrama de clases de la aplicación en RPA

- ***Main***: clase desde la que comienza a ejecutarse la lectura de datos y generación de sensores.
- ***Sensor***: representa un sensor que comienza a leer la información por el puerto físico al que está conectado, la cual es procesada para comprobar su validez y almacenada una vez ratificada.
- ***Stream***: flujo de datos que cumple con la configuración establecida con respecto a la lectura de los sensores asociados, la periodicidad de envío de la información, fecha de validez y dirección de destino.
- ***Forward***: crea el socket UDP necesario a la máquina de destino así como el datagrama donde se incluye la información que se desea enviar a un determinado *host* y puerto.

- ***SensorValue***: contiene los datos leídos por el sensor con la sintaxis requerida por el usuario, además de su correspondiente marca de tiempo.

A.1.2. Aplicación del GCS

La aplicación instalada en la Estación de Control de Tierra es la que funciona como servidor y garantiza el acceso al usuario a la configuración de flujos y sensores. En la figura A.2 se muestran las cuatro clases Java principales de ésta aplicación, cuyas definiciones son las siguientes:

- ***Controller***: es el servidor web, permite el acceso a la aplicación a través de un navegador.
- ***ConnectionSSH***: crea la conexión SSH entre el servidor y el RPA para el envío y recepción del fichero XML de configuración a través de SFTP.
- ***CreateXML***: clase principal de la Entidad de Gestión, se encarga de la manipulación de los *arrays* de sensores y flujos.
- ***Abnf***: manipula el fichero ABNF alojado en el servidor, lo compila y comprueba las reglas sintácticas introducidas por el usuario.

Al igual que en la aplicación del RPA, se ha creado un paquete denominado **beans** donde se almacenan las clases destinadas a la manipulación de los objetos creados a partir de los elementos XML del fichero de configuración.

A.2. Esquema XML

La configuración de todos y cada uno de los sensores y flujos creados en la aplicación es almacenada en un único fichero. Para asegurar que la información guardada es correcta, se hace uso de un esquema en el que se define la apariencia del fichero de configuración, es decir, se define qué elementos puede contener, en qué orden y cuál puede ser su contenido.

Así pues, tal y como se puede ver en la figura A.3, la estructura del fichero de configuración consiste en un elemento principal, **data**, y los subelementos **sensor** y **stream**. Éstos elementos contiene a su vez otros subelementos, y así, hasta que un subelemento **period** contiene un valor en lugar de más subelementos. De esta forma se distingue entre elementos complejos (**complexType**), aquellos que

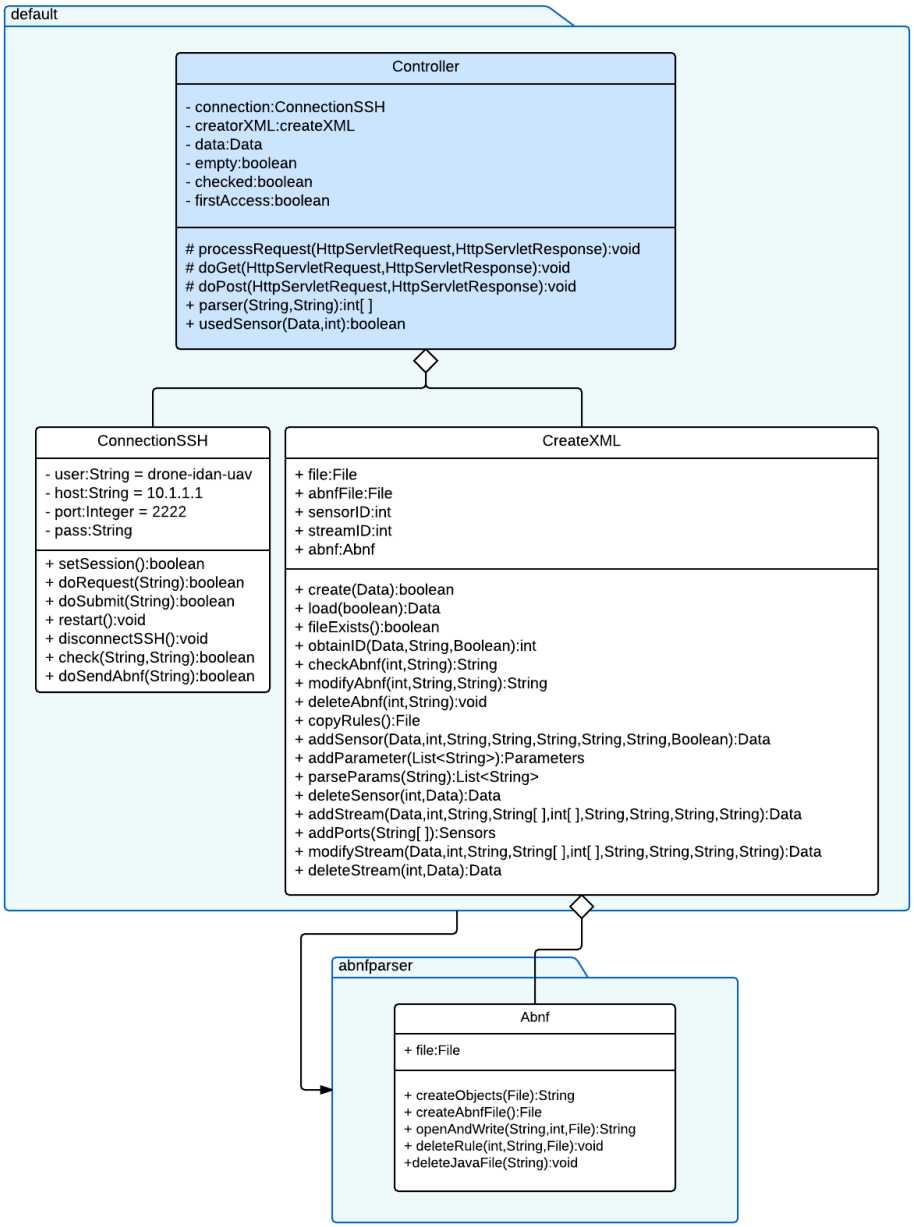

Figura A.2: Diagrama de clases de la aplicación en GCS

contienen subelementos y elementos simples (**simpleType**), que no contienen otros subelementos.

Los elementos complejos que aparecen en el esquema son los siguientes:

- ***data***: elemento raíz.

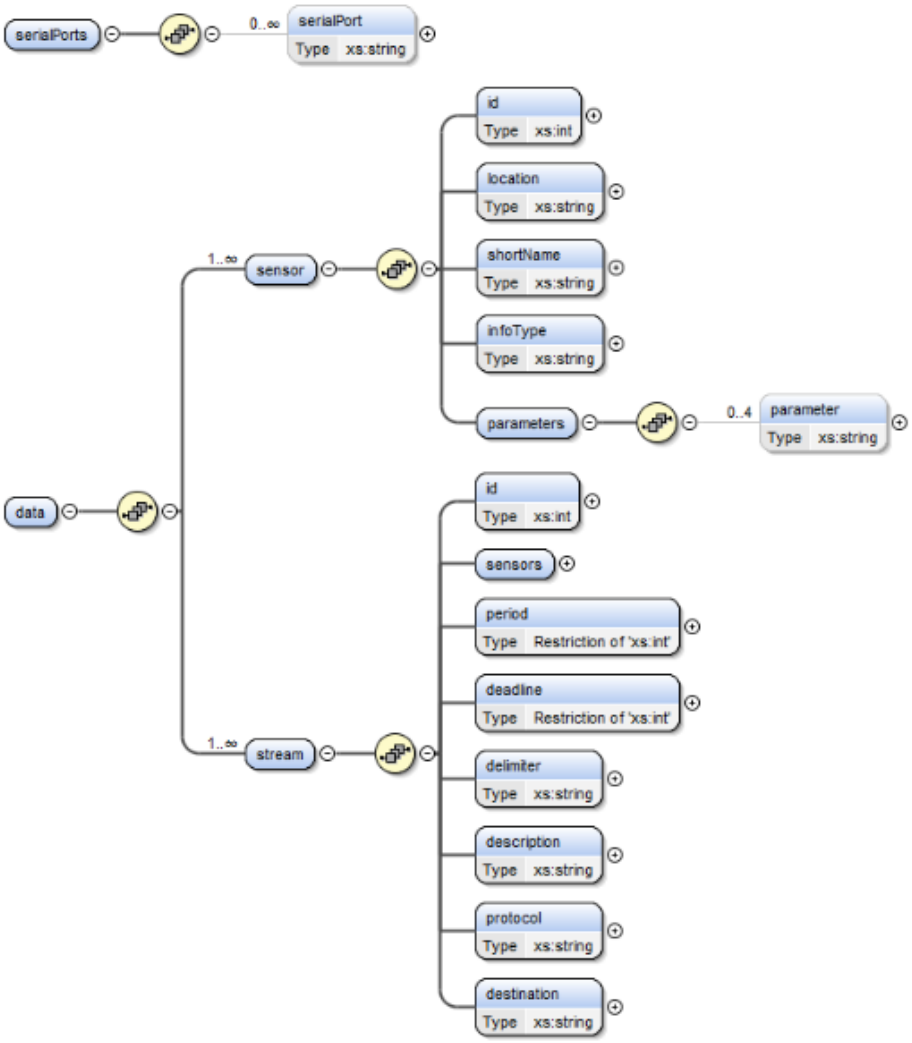

Figura A.3: Esquema XML del fichero de configuración de flujos y sensores

- ***sensor***: elemento que puede ser replicado un número ilimitado de veces ya que el indicador **maxOccurs** tiene el valor **'unbounded'**. Define un sensor.
- ***parameters***: define los cuatro parámetros de configuración del puerto serie, por ello su aparición está restringida entre cero y cuatro veces.
- ***stream***: al igual que el elemento *sensor*, su número de apariciones no está limitado. Define un flujo de datos.
- ***sensors***: define el listado de sensores en los que cada flujo está registrado. Debido a que el número de sensores que se pue-

den configurar es ilimitado, el tamaño de este listado tampoco está restringido.

El resto de elementos que componen el esquema son de tipo simple, por lo que solamente es necesario definir su tipo de datos y las restricciones que lleven asociadas. En este caso, los únicos elementos que tiene restricciones son *period* y *deadline*, ya que deben recibir valores mayores a cero para que la configuración del flujo de datos sea correcta.

Anexo B

Descripción de la maqueta

En este anexo se presenta una enumeración detallada de cada uno de los elementos que componen la maqueta utilizada en las pruebas de validación del sistema.

- El **router UAV** ha sido construido a partir de un sistema PC/100 rugerizado, con un procesador Intel Core 2 Duo a 1.86 GHz, 2 GB de memoria y 4 GB de disco de memoria flash. El i incluye 2 puertos GbE (Gigabit Ethernet) que permiten la conexión con los subsistemas de LOS y satélite además de un switch PCIe/Ethernet con 4 puertos GbE que pueden ser utilizados para conectar la IP de los distintos componentes disponibles en el RPA (el número de puertos disponibles puede ser incrementado conectando tantos switches PCIe/Ethernet como sean necesarios).

- El **router GCS** ha sido desplegado en un ordenador con un diseño específico que consta de un procesador AMD Athlon II X2 a 3 GHz, 8 GB de memoria, 500 GB de disco duro y 3 puertos GbE que permiten la conexión con los susbistemas LOS y satélite en el GCS. Este router provee un enlace de datos a redes IPv6 externas y utiliza Tayga [42], una implementación de NAT64 fuera de núcleo (en inglés,*out-of-kernel*). Tanto el router UAV como el router GCS tienen instalado el sistema operativo Debian GNU/Linux 6.0.6 i686. Además en ambos se ha habilitado seguridad IPsec mediante el paquete Debian *ipsec-tools* 0.7.3 [54].

- Un prototipo de un **sistema RVT** (del inglés *Remote Video Terminal* o Terminal de video remoto) es implementado utilizando un ordenador portátil con procesador Intel Core 2 Duo a 2.4 GHz y 2 GB de memoria, donde el sistema operativo

utilizado es el mismo que el instalado en los routers GCS y RPA. Asimismo, este componente incluye soporte para IPsec y el sistema de máquinas virtuales VirtualBox [53], utilizado para ejecutar una versión virtualizada de la estación de control instalada en el GCS. De esta forma, la telemetría puede ser procesada y visualizada por un usuario desde este RVT.

Appendix C

Introduction

This chapter explains the main concepts behind the project, those which motivated the development of the system. Next, it is detailed the aims to be reached and finally, it is introduced the dissertation structure.

C.1. Motivation of the project

Today, talking about drones or UAVs (*Unmanned Aerial Vehicle*) is talking about the new idea of nature monitoring, security and military missions, among other things.

A UAV can have multiple ranges (tactical, MALE or HALE) and capabilities, and it is dependent of a ground control station (GCS). Moreover, each unmanned aerial vehicle is able to carry different payloads depending on the mission carried out by the whole system. In that way, a UAV may have integrated a temperature sensor, infrared or video cameras, synthetic aperture radars, RFID (Radio Frequency Identification) sensors or another electronic device. The generated sensor information in the UAVs is collected and processed by an acquisition data unit which sends this telemetry information to the GCS, from where it is retransmitted to the destinations in charge of analysing the data.

Once the working elements that integrates the UAV have been introduced, it is interesting to point out which are their applications. Within the military field, the use of these vehicles has meant a revolution in terms of search and rescue missions, disaster prevention and management, and the supply of a 24-hour "eye in the sky" in those areas whose access is not easy with manned flight.

In addition to the large benefits that bring along with the use of UAVs in the battlefield, there is also plenty of surprising civilian

uses related with UAV technology since it has been made public, though different media, to the people who have become aware of its utility. This fact and technology evolution is making available the utilization of RPAs in hurricane hunting, 3D mapping, protecting wildlife and SAR (search and rescue) missions.

The graph of the figure C.1 shows a forecast of how the number of UAVs in the airspace of the United States is expected to evolve in the next 20 years, according to the Department of Transportation of the USA. As it can be seen in the figure, in 2035 the number of UAVs is expected to be almost 180.000 only in one country; that data reveals how RPA field is about to explode yet and becomes the main motivation of developing a project like the one introduced in this dissertation.

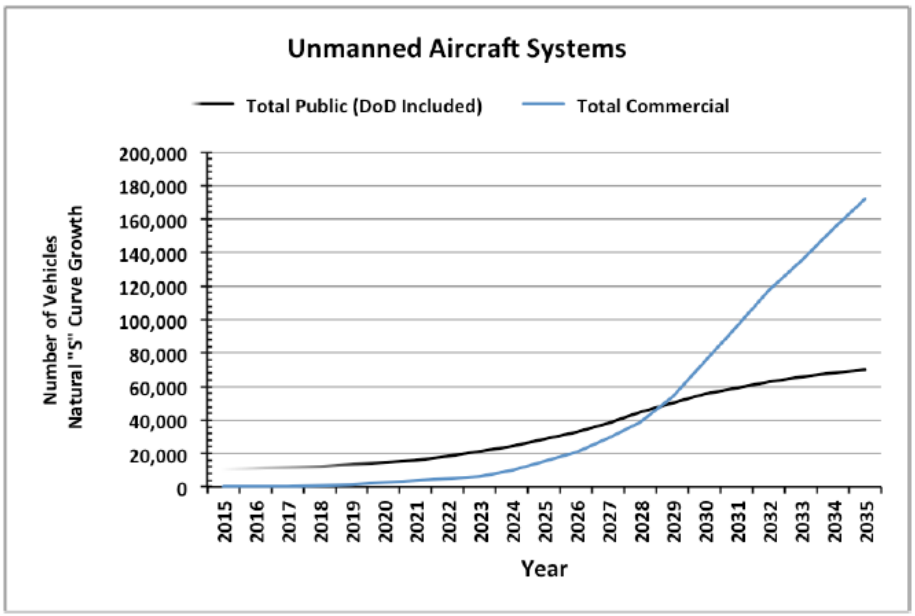

Figure C.1: Total UAS Forecast 2015 - 2035[1]

In the future, these miles of UAVs will produce an enormous amount of information that will need to be manipulated through software and hardware equipment. Nowadays, market-based solutions are private and they have high costs besides their difficult management and updating. Examples might be the systems introduced by Ixion Industry and Aerospace [1] and the French shipbuilder company DCNS [19].

In this context is developed the current dissertation, whose aim

[1] Archive. Department of Transportation of the USA Government. https://fas.org/irp/program/collect/service.pdf

is to design and implement a low cost, software platform for configuring and executing collecting, processing and sensor data sending tasks, in such a way that it should be extendible and flexible enough to be able to integrate any sensor type.

Finally, it is important to point out that this project is a future work line that is identified at the end of the DRONE project (DN8644-COINCIDENTE-10032110042), which was funded by the Ministry of Defence and which was developed jointly by the Instituto Nacional de Técnica Aeroespacial (INTA) and the University Carlos III de Madrid.

C.2. Aims

In this project there is a main objective which is the design and development of a platform for collecting and sending sensor information of a UAV system. This primary objective can be divided into several sub-goals like the ones introduced next:

- The system is based on current Internet protocols and standards.
- The platform supports the process execution for collecting, encapsulating and sending of sensor information from a UAV to multiple destinations.
- The provided service must be flexible with a view to accommodating the information collection of a variety of sensors with different natures and capabilities.
- It is looked for a cost-efficient system, because of that, it is developed through the use of open source software tools.
- The platform is entirely configurable by the user through a management web interface, which is expected to support the flexible configuration of the processes for the collection, encapsulation and sending of sensor information from the managed UAVs.
- The proper functioning of the system is expected to be tested over a working model of UAS, which is available at the Telematics Engineering Department of the University Carlos III of Madrid.

C.3. Structure of the document

The document is divided into seven chapters and five appendices, which are the following:

- The current chapter, **Introduction**, covers those aspects related to the context in which the project is developed, the goals to achieve and the description of the content organization.
- Chapter 2, **State of the art**, makes a review of the RPAs history and presents the different types of these vehicles. Apart from that, the current technologies connected to web applications development are described.
- Chapter 3, **Analysis and system design**, examines the scenario in which the system has to work. First, a description of the scheme and the initial model is introduced and later, the requirements for each of the modules that integrate de system are enumerated.
- In chapter 4, **Software implementation**, a general approximation of the chosen solution is given. After that, the main use cases and procedures are explained in detail.
- Chapter 5, **Acceptance testing**, lists those functionalities covered by the implemented applications. Moreover, it shows the results of checking the running of the system by means of different graphs.
- Chapter 6, **Project planning**, describes the organization of the whole project in terms of work management and cost analysis.
- In chapter 7, **Conclusions and future work**, issues like difficulties found during the development of the project and the results obtained are commented. In addition, future improvements and lines of work are introduced.
- Appendix A, **Class diagrams and XML schema**, includes the class diagrams of the two applications of the system and a explanation of the main configurable elements.
- Appendix B, **Model description**, where the description of the main working model components is introduced.
- In Appendix C, **Tntroduction**, the English translation of the chapter 1 is introduced.

- Appendix D, **Extended summary**, presents a extended summary oh the project dissertation in English.
- In Appendix E, **Conclusions and future work**, the English translation of the chapter 7 is introduced.

Appendix D

Extended summary

D.1. Introduction

Today, talking about drones or UAVs (*Unmanned Aerial Vehicle*) is talking about a new idea of nature monitoring, security and military missions, among other things.

There is plenty of surprising civilian uses related with UAV technology since it has been made public, though different media, to the people who have become aware of its utility. This fact and technology evolution is making available the utilization of RPAs in hurricane hunting, 3D mapping, protecting wildlife and SAR (search and rescue) missions.

The graph of the figure C.1 shows a forecast of how the number of UAVs in the airspace of the United States is expected to evolve in the next 20 years, according to the Department of Transportation of the USA. As it can be seen in the figure, in 2035 the number of UAVs is expected to be almost 180.000 only in one country; that data reveals how RPA field is about to explode yet and becomes the main motivation of developing a project like the one introduced in this dissertation.

In the future, these miles of UAVs will produce an enormous amount of information that will need to be manipulated through software and hardware equipment. Nowadays, market-based solutions are private and they have high costs besides their difficult management and updating.

All in all, the main aim is to design and implement a low cost, software platform for configuring and executing collecting, processing and sensor data sending tasks, in such a way that it should be extendible and flexible enough to be able to integrate any sensor

[1] Archive. Department of Transportation of the USA Government. https://fas.org/irp/program/collect/service.pdf

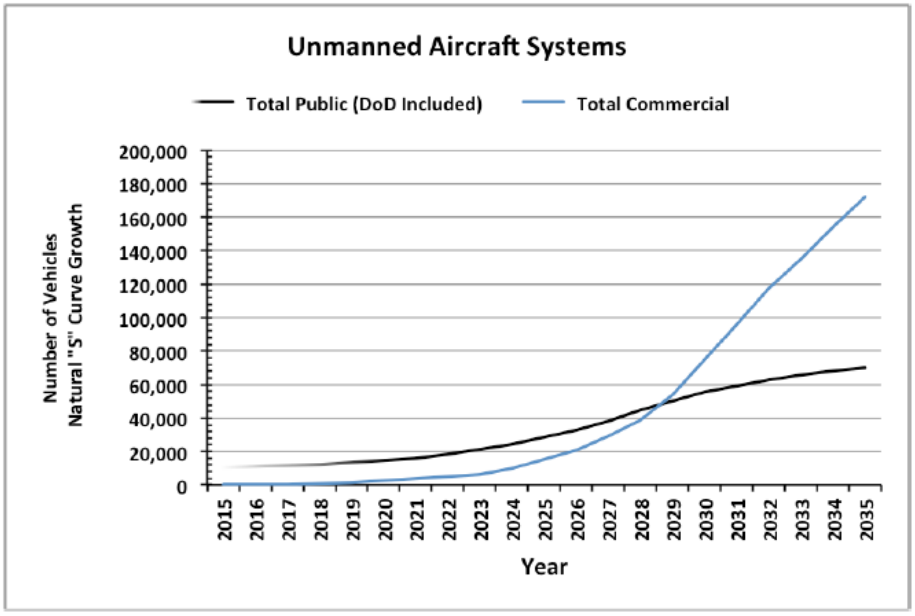

Figure D.1: Total UAS Forecast 2015 - 2035[1]

type..

D.2. Design overview

The setting for this project consists on the typical RPA system where three main components can be found: the RPA unit, the ground control station and the group of users who wants to access the application.

The RPA that is being used is the SIVA, an unmanned aerial vehicle created by the INTA (the Spanish Aerospace Technology National Institute) and funded by the Spanish Ministry of Defence. Each unmanned plane has installed different sensors which generate information that is collected by a data acquisition unit. This element sends the telemetry data to the GCS from where it is retransmitted to the concerned receivers.

The figure D.2 shows how the components that make up the system are related with each other. Over that model, this project looks for developing a sensor information management system where different users can access it in order to configure how, when and what is received.

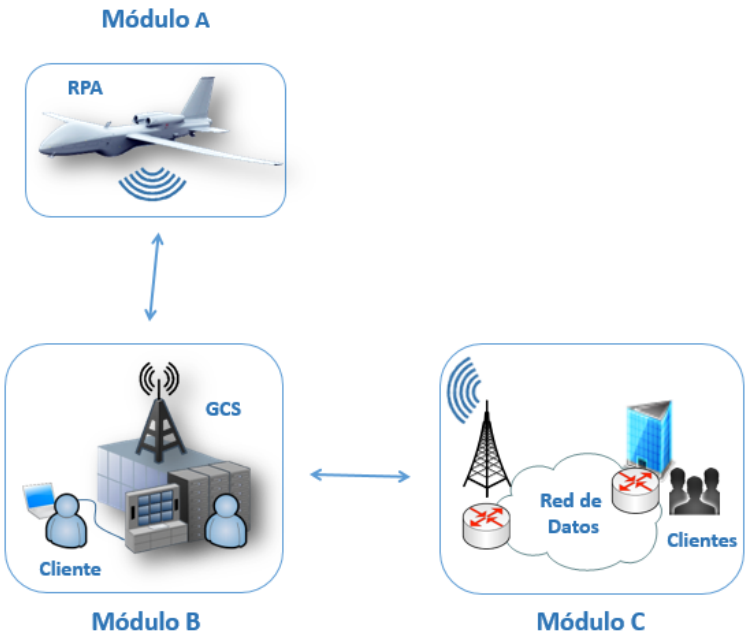

Figure D.2: Module representation of the system scheme

D.3. System implementation

The system development has been carried out in three stages, each one dedicated to the implementation of one of the funcional modules that compose the whole system.

In the next figure we can see the different technologies used in each of the modules as well as the protocols through which the communication between these three components is established.

The user access and server modules consists on a single application and the RPA module consists on another one. Thereupon a brief description of each module is introduced:

- **User access module**: the application graphical user interface (GUI) provides access to the user, through which it is able to authenticate itself into the system, choose the language or manage sensors and streams. The GUI is accessible via a web platform that asks for the data to the server module. Moreover, this GUI allows the user to interact with the web page contents. The communication with the server is made through HTTP protocol.
- **Server or gateway module**: this module is the responsible for communicating the RPA and the user. On the one hand, it

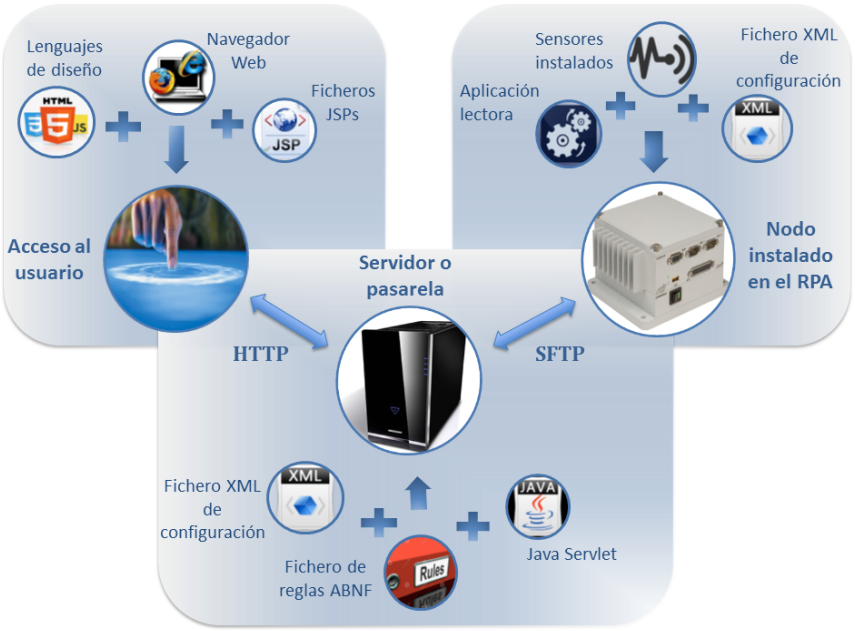

Figure D.3: System modules connection

serves the web pages to the client browser. On the other hand, it is in charge of converting the data introduced by the user into XML elements. The communication between this module and the RPA is made through SFTP protocol, which sends the XML configuration file in a secure way to the RPA.

- **RPA module**: this module is an application on its own which is dedicated to manage sensor information. Due to that, there should be a copy of this application in every RPA unit that is a member of the RPA system. Again, the communication between this module and the server is given through SFTP. The way it works is the following: the application reads sensor information and creates data streams according to the configuration saved in the XML file, then, in a certain moment, the data read and stored in the streams is sent to the concerned receivers though UDP packets.

D.4. Acceptance testing

The system functionality has been tested through a usability evaluation of the web platform, which have led to the following conclusions:

- The user is able to change the interface navigation language, besides, he can return in any moment to the starting view in order to change the language.
- The two main configurable elements are easily recognizable and the user is able to switch between both configurations without problems.
- When the user introduce unexpected data, error messages are thrown together with a brief explanation about the correct data type expected by the application.
- The user is able to check at any moment that it is logged in the application, since its name appears in every view of the web page during the session.
- The user can add, delete or modify any stream or sensor if and only if the necessary conditions in the case of the data stream deletion is fulfilled.
- The browsability through the interface is simple because the information is shown concisely so as not to confuse the user.
- The accessibility of the web platform for people with reduced vision problems is born since it is used a large lettering marked and a high contrast between the background colour and the font colour.

In addition to the previous evaluation a set of tests have been done over a working model already built owned by the Telematics Engineering Department of the University Carlos III of Madrid. Next, one of these tests is introduced just to make clear how the validation is done.

At the beginning, several sensors are already installed but no stream is configured, so when the first two data streams are created, this is the resulting graph that is obtained with a network analyser: As it can be seen in the above figure, the streams have different periodicity, the red one appears each second while the black one is sent each 2.5 seconds.

The remainder validation tests (for checking how the stream modification and elimination works) follow the same guideline commented previously. For that reason, and due to the fact that the current appendix is just a summary, those tests are only included in the complete dissertation.

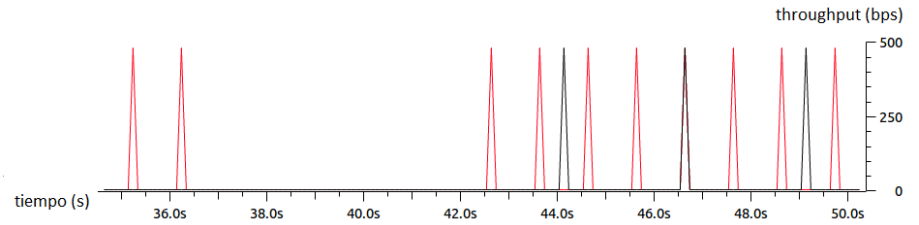

Figure D.4: Resulting graph of adding two streams

D.5. Conclusions

This project was oriented to develop the most flexible sensor configuration system that were possible. To do that, two applications divided into three modules connected between them were deployed. The system modularization has allowed to integrate the technologies required in such a way that all of them work together. Moreover it has simplified error detection as well as the development of future extensions of the project.

It is important to point out that the system has been tested with few sensors, but the limitation of the number of this sensors is determined by the physical serial ports available in the RPA device.

Regarding the project difficulty, it must be said that at the beginning, using unexplored technologies such as ABNF grammar and XML schema seemed to be complicated. However, the initial idea of dividing the development of the project in slight modules was entirely appropriate because it made easier the implementation of the whole system and the integration of the different elements of which it is composed, too.

Finally, the last big issue takled was the research of a software which made compatible Java language with serial ports. The first thought was to use the Java Communications API but it is no longer supported, so the alternative was RXTX, an API that has allowed get a working communication between physical sensors and the application running in the RPA, without any problem.

To sum up, after all these months of work, it is considered that the global result obtained is satisfying, so the main goal of creating a **flexible** system, capable of reading data with a concrete syntax and sending this information with a changeable rate has been achieved.

D.5.1. Future work

The new lines of work related to the project are the following:

- **New serial connection support**

  Currently, the software connection between the sensors and the RPA is made through a Java library called RXTX, which allows to the RPA application reading from the sensors installed in it.

  Since Java version 8, the library above commented is no longer supported without doing some modifications; for this reason, in the future, it will be necessary to update the API utilized for communicating the sensors with the RPA.

  One of the libraries for working with serial port which is much utilized now is the so called jSSC (*Java Simple Serial Connector*). This open source project is characterized for its ease, so the idea of this working line would be to adapt the implementation of the RPA application to use this library and consequently, avoiding the obsolescence of the whole system.

- **New transport protocols**

  The implemented system should fast transmit the data read from the sensors, for this reason, UDP is the selected transport protocol.

  However, in the future, the user could look for guaranteed delivery of the transmitted data, and the UDP protocol used until now would not be valid.

  Given the modularity of the system developed in this project, TCP and UDP could be both used. To do so, it would be necessary to implement another Java class with the pertinent methods to create TCP packets and send them.

- **Data forwarding system**

  In the future, the situation of two clients, one operating from de GCS and another one working from a network external to the GCS, might arise. With the actual system, if both clients connects to the same RPA, the applications running on it will send twice the data, which in not the best performance.

  The idea to be developed for avoiding this situation is to design a system in the server module that forwards the information received in the GCS from it to all the clients connected from external networks.

- **Time optimization**

  The current waiting time of the RPA application when a configuration change is made is too much longer.

Because of that, one of the ideas to be implemented in the future is the optimization of this restarting time which depends on the compilation and reexecution of the program running in the RPA. Knowing that the java command for compiling files, **javac**, does not have any option to optimize the compilation, what could be done is to apply the java interpreter with options like *-Xmixed* when executing the program, which makes that repetitive code is executed more efficiently.

- **Security support**

  Currently, encryption of data sends from the RPA to the interested receivers is just guaranteed in the IP and link layer though IPsec tunnels. Because of that, it would be interesting to have a tool that provides security support at the application level such as only source and destination hosts are able to access the collected sensor data.

Appendix E

Conclusions and future work

In this chapter, general conclusions about the project are presented as well as the difficulties found during its design and implementation. Besides, the obtained results are commented and finally, a set of ideas for future work in this system is introduced.

E.1. Conclusions about the project

The developing of this project was oriented to achieve the most flexible sensor configuration system that were possible. To do that, several technologies were integrated so that they all work together.

The integration of all the components is reduced to the deployment of two applications divided into three modules which are connected to each other. The idea of modularizing the system allows to simplify the error detection as well as facilitates future extensions of the project.

It is important to point out that the system has been tested with few sensors. However, the limitation of the number of this sensors is determined by the physical serial ports available in the RPA device and not by the application.

E.1.1. Project development

The project development has been carried out in modules. First, the module related with the client side was made since it was necessary to have it developed in order to be able to interact with the application.

After that, the logic behind the server module and the server web itself were built. In this stage, unknown languages such as ABNF

grammar and XML schema were used. Before making available the connection between server and RPA modules, it was necessary to study how ABNF works and how it should be integrated in the project.

At the beginning, using these unexplored technologies seemed to be complicated. However, the initial idea of dividing the developing of the project in slight modules was entirely appropriate because it made easier the implementation of the whole system and the integration of the different elements of which it is composed, too.

Within the development stage of this module, one of the greatest challenges to overcome was the interrupts suffered when the server had to connect via SSH to the RPA. In order to guarantee the connection security between both modules, the SSH server (installed on the RPA) had activated password-based authentication, reason why every time that it was asked for accessing into the remote RPA, the user id was required so that the access was granted and the program execution could continue.

To avoid this problem it was decided to use public-key based authentication; by introducing that change, security was improved an breaks were suppressed since SSH server does not ask for user id in order to grant it the access, due to the SSH server has stored the public key shared between them.

Later, when the functional validations were performed, some challenges appeared about the sensors since there were not enough physical ones that could be connected to the available RPA serial ports for checking the application performance. This problem was solved by creating several logic ports each one symbolic linked to a serial port, which data were constantly written in.

Finally, the last big issue tackled was the research of a software which made compatible Java language with serial ports. The first thought was to use the Java Communications API but it is no longer supported, so the alternative was RXTX, an API that has allowed get a working communication between physical sensors and the application running in the RPA, without any problem.

One this was overcome, the rest of the project development was manageable thanks to the wealth of information associated with Java, since there is always a library for implementing the functionality that has been looked for.

E.1.2. Achieved result

After all these months of work, despite the difficulties of working with techonologies on which there was not a former knowledge, the

aim of creating a **flexible** system, capable of reading data with a concrete syntax and sending this information with a changeable rate, between other characteristics, has been achieved.

The views on the client side are both esthetic and intuitive. It is true that there could be some helping windows to guide the user throught the interface, because for now, there are only error messages thrown when the user introduces unexpected data.

The performance of the remainder components of the system is correct and although the waiting time bewteen changes in the configuration could be improved, it is considered that the global result obtained is satisfying.

E.2. Future work

In this section, new lines of work related to the project are mentioned.

E.2.1. New serial connection support

Currently, the software connection between the sensors and the RPA is made through a Java library called RXTX, which allows to the RPA application reading from the sensors installed in it.

Since Java version 8, the library above commented is no longer supported without doing some modifications; for this reason, in the future, it will be necesary to update the API utilized for communicating the sensors with the RPA.

One of the libraries for working with serial port which is in fashion is the so called jSSC (*Java Simple Serial Connector*). Despite this open source project is not stable yet, it is characterized for being simple, which make it possible to migrate from RXTX to this API without many problems. Because of that, the idea of this working line would be to adapt the implementation of the RPA application to use this library and consequently, avoinding the obsolescence of the whole system.

E.2.2. New transport protocols

The UDP protocol is a fast way to transmit data because the sender will not wait to make sure the receipient received the packet. Due to the fact that the implemented system should fast transmit the data read from the sensors, this has been the selected transport protocol.

However, in the future, the user could look for guaranteed delivery of the transmitted data, and the UDP protocol used untill now would not be valid since it does not provides flow control or error correction.

A potentially interesting working line related with it, is the incorporation of new mechanisms in charge of information delivery from RPA units. As an example, TCP support could be added in order to provide a reliable connection-oriented service. It is also interesting to consider the new protocols which are currently developing the emerging scenarios of the Internet of Things such as CoAP [64] or those mechanisms that are centered in the information (e.g. Data-Distrubtion Service – DDS [1]).

Related with the posibility of using TCP, it is necessary to point out that the RPA application would need to be modified in order to integrate this transport protocol. Currently, there is just one Java class called ***Forwarded*** in charge of sending the information originating from the sensors through UDP to the receiver. For that reason, in order to add a new transport protocol it would just be necessary to implement another Java class with the pertinent methods to create TCP packets and send them.

E.2.3. Data forwarding system

In the future, the situation of two clients, one operating from de GCS and another one working from an external network (called *"Centros de control externos"* in the figure 3.1 of the chapter 3) to the GCS, might arise. With the actual system, if both clients connects to the same RPA, the applications running on it will send twice the data, which in not the best performance.

The idea to be developed for avoiding this situation is to design a system in the server module that forwards the information received in the GCS from it to all the clients connected from external networks. By using this way of procedure, the information is only sent one time form the RPA.

E.2.4. Time optimization

As it can be seen in chapter 5, when a change is applied in the configuration of any sensor or stream, the applicaction deployed in the RPA is restarted. Currently, the restarting time is about 6 seconds which is a bit longer than the desired delay.

[1] Data Distribution Service Portal. http://portals.omg.org/dds/

Because of that, one of the ideas to be implemented in the future is the optimization of this restarting time which depends on the compilation an reexecution of the program running in the RPA. Knowing that the Java command for compiling files, **javac**, does not have any option to optimize the compilation, what could be done is to apply the Java interpreter with options like *-Xmixed* when executing the program, which makes that repetitive code is executed with more efficacy.

On the other hand, a study of other java compilers could be done to determine if there is any one that could give a better performance than the obtained untill this moment.

E.2.5. Security support

Nowadays, encryption of data sends from the RPA to the interested receivers is just guaranteed in the IP and link layer though IPsec tunnels[2]. Although the idea of hiding the delivered information is not whithin the scope of this project, it is interesting like future line of work to have a tool that provides security support at the application level such as only source and destination hosts are able to access the collected sensor data.

[2]IPsec is deployed over the working model used for developing and validating the implemented platform, which is available at the Telematics Engineering Department of the University Carlos III of Madrid.

Bibliografía

[1] IXION Industry & Aerospace. Sistemas de control, navegación y misión (cnm) para vehículos no tripululados. http://www.ixion.es/es/productsservices/UnmannedVehicles.html, 2013. Último acceso: 10/06/2015.

[2] Arduino. Arduino ethernet. http://www.arduino.cc/en/Main/ArduinoBoardEthernet, 2015. Último acceso: 08/06/2015.

[3] Patricio Martínez Barco Armando Suárez Cueto. Sistemas de gestión de bases de datos, 2007. Último acceso: 23-04-2015.

[4] Unmanned Aerial Vehicle Systems Association. `https://www.uavs.org`, 2015. Último acceso: 26-03-2015.

[5] Reg Austin. *Unmanned aircraft systems : UAVS design, development and deployment*, chapter 1. John Wiley & Sons Ltd, 2010.

[6] Berenice Backer. Uav evolution – how natural selection directed the drone revolution. `http://www.army-technology.com/features/featureuav-evolution-natural-selection-drone-revolution/`, 2012. Último acceso: 27-03-2015.

[7] Stig Saether Bakken, Zeev Suraski, and Egon Schmid. *PHP Manual: Volume 2*. iUniverse, Incorporated, 2000.

[8] Daniel Braun Tim Hoffmann Benito van der Zander, Jan Sundermeyer. Página web texstudio. http://texstudio.sourceforge.net/, 2015. Último acceso: 30/05/2015.

[9] Aldo Bongio, Piero Fraternali, Marco Brambilla, Sara Comai, and Maristella Matera. *Morgan Kaufmann series in data management systems: Designing data-intensive Web applications*. Morgan Kaufmann, 2003.

[10] Roland Bourret. Xml Data Binding Resources. http://www.rpbourret.com/xml/XMLDataBinding.htm, 2011. Último acceso: 27-04-2015.

[11] BuiltWith®. Javascript Usage Statistics. `http://trends.builtwith.com/javascript`, 2015. Último acceso: 16-04-2015.

[12] T. Goff J. H. Kim J. Macker J. Weston N. Neogi A. Ortiz C. Danilov, T. R. Henderson and D. Uhlig. Experiment and field demonstration of a 802.11-based ground-uav mobile adhoc network. In *Military Communications Conference*, pages 1–7. IEEE, 2009.

[13] D. Connolly and L. Masinter. The 'text/html' media type. RFC 2854, RFC Editor, June 2000. `http://www.rfc-editor.org/rfc/rfc2854.txt`.

[14] Worl Wide Web Consortium. Css current work and how to participate. `http://www.w3.org/Style/CSS/current-work`, 2015. Último acceso: 16-04-2015.

[15] Oracle Corporation. Página web netbeans. https://netbeans.org/, 2015. Último acceso: 30/05/2015.

[16] Yokogawa Electric Corporation. Mw100 data acquisition unit. http://tmi.yokogawa.com/products/data-acquisition-equipment/low-speed-daq-industrial-recorders/mw100-data-acquisition-unit/, 2015. Último acceso: 10/06/2015.

[17] D. Crocker and P. Overell. Augmented bnf for syntax specifications: Abnf. STD 68, RFC Editor, January 2008. `http://www.rfc-editor.org/rfc/rfc5234.txt`.

[18] CWC-AE. Acra kam-500 airborne data acquisition unit (dau). http://www.cwc-ae.com/data-acquisition-concentrators/data-acquisition, 2015. Último acceso: 10/06/2015.

[19] DCNS. Naval integration of observation drones. http://en.dcnsgroup.com/activities/products/#integration-navale-drone-observation, 2015. Último acceso: 10/06/2015.

[20] Notas de prensa del SENER. Ga-asi y sener se alían para promocionar el predator b en españa. http://www.sener.es/News/ga-y-sener-se-alian-para-promocionar-el-predator-b-en-espana/es#.VTUD8ZNKW1o, 2015. Último acceso: 20-04-2015.

[21] Instituto Nacional de Técnica Aeroespacial .Esteban Terradas". SIVA. Air Surveillance System. 2012.

[22] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, RFC Editor, August 2008. `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[23] W3C España. Guía breve de tecnologías xml. http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML, 2015. Último acceso: 27-04-2015.

[24] Norberto Fernández. Almacenamiento y gestión de información(i).

[25] Norberto Fernández. Lenguajes de marcado HTML, XML. 2013.

[26] Apache Software Foundation. Oficial web of apache tomcat. http://tomcat.apache.org/, 2015. Último acceso: 14/05/2015.

[27] Linux Foundation. Linux foundation and leading technology companies launch open source dronecode project. http://www.linuxfoundation.org/news-media/announcements/2014/10/linux-foundation-and-leading-technology-companies-launch-open, October 2014. Último acceso: 09-04-2015.

[28] Wireshark Foundation. Página web de wireshark. https://www.wireshark.org/, 2015. Último acceso: 30/05/2015.

[29] Brian Fung. Drones don't just fly. here's one that crawls inside sewer pipes. http://www.washingtonpost.com/blogs/the-switch/wp/2013/08/13/drones-dont-just-fly-heres-one-that-crawls-inside-sewer-pipes/, 2013. Último acceso: 06-04-2015.

[30] Jesse James Garrett. Ajax: A New Approach to Web Applications. *Adaptive Path*, 2005.

[31] Hibernate. Official web of hibernate tool. http://hibernate.org/, 2015. Último acceso: 27-04-2015.

[32] B. Hoehrmann. Scripting media types. RFC 4329, RFC Editor, April 2006. `http://www.rfc-editor.org/rfc/rfc4329.txt`.

[33] National Instruments. Compactdaq. www.ni.com/data-acquisition/compactdaq/esa/, 2015. Último acceso: 10/06/2015.

[34] Augusto Iturri. Drones y agricultura: Los aviones no tripulados podrían revolucionar la producción. http://www.drones-argentina.com.ar/2013/05/22/drones-y-agricultura-los-aviones-no-tripulados-podrian-revolucionar-la-produccion/, 2013. Último acceso: 06-04-2015.

[35] Robert J.Brunner. *JSP. Practical Guide for Java Programmers.* Morgan Kaufmann, 2003.

[36] JCraft. Jsch - java secure channel. http://www.jcraft.com/jsch/, 2014. Último acceso: 30/05/2015.

[37] Rajesh Kumar. *Tactical Reconnaissance: UAVs versus Manned Aircraft.* BiblioScholar, 2012.

[38] Seth Ladd. *Expert Spring MVC and Web Flow.* Springer, 2010.

[39] S. Lehtinen and C. Lonvick. The secure shell (ssh) protocol assigned numbers. RFC 4250, RFC Editor, January 2006. `http://www.rfc-editor.org/rfc/rfc4250.txt`.

[40] Hakon Lie, Bert Bos, and Chris Lilley. The text/css media type. RFC 2318, RFC Editor, March 1998. `http://www.rfc-editor.org/rfc/rfc2318.txt`.

[41] Per Ohlckers Luca Petricca and Christopher Grinde. Micro- and nano-air vehicles: State of the art. *International Journal of Aerospace Engineering*, 2011.

[42] Nathan Lutchansky. Tayga, a stateless nat64 implementation for linux. http://www.litech.org/tayga/, 2011. Último acceso: 10/06/2015.

[43] D. Hampel M. F. Pinkney and S. DiPierro. Unmanned aerial vehicle (uav) communications relay. In *Military Communications Conference Proceedings*, volume 1, pages 47–51. IEEE, 1996.

[44] C. Bettstetter G. Friedrich H. Hellwagner B. Rinner M. Hofbaur M. Quaritsch, E. Stojanovski and M. Shah. Collaborative microdrones: applications and research challenges. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, page 38, 2008.

[45] D. Wischounig-Strucl S. Bhattacharya M. Shah M. Quaritsch, K. Kruggl and B. Rinner. Networked uavs as aerial sensor

network for disaster management applications. *Elektrotechnik und Informationstechnik*, 127(3):56–63, 2010.

[46] H. Hellwagner-B. Rinner A. Adria M. Quaritsch, R. Kuschnig and U. Klagenfurt. Fast aerial image acquisition and mosaicking for emergency response operations by collaborative uavs. In *Proceedings of the 8th International ISCRAM Conference-Lisbon*, volume 1, 2011.

[47] Luis Fernando Real Martín. Introducción a los uav y una visita al salón de seguridad y defensa ́homsec13*: Revista Antena. Colegio Oficial de Ingenieros Técnicos de Telecomunicación*, 2013.

[48] Matt McFarland. In switzerland, police find a use for drones. *The Washington Post*.

[49] Microsoft. Web oficial de ms windows.

[50] Sergio Luján Mora. *Programación en Internet: Clientes WEB*. Editorial Club Universitario, 2001.

[51] Inc. npm. List of jquery plugins. `https://www.npmjs.com/browse/keyword/jquery-plugin`, 2015. Último acceso: 16-04-2015.

[52] Oficial Web of IETF. The internet engineering task force. https://www.ietf.org/. Último acceso: 11-05-2015.

[53] Oracle. Vm virtualbox. https://www.virtualbox.org, 2015. Último acceso: 10/06/2015.

[54] package ipsec tools. Ipsec utilities for linux. http://packages.debian.org/squeeze/ipsec-tools, 2015. Último acceso: 10/06/2015.

[55] parse2. aparse. http://www.parse2.com/index.shtml, 2014. Último acceso: 30/05/2015.

[56] SAX project. Official website for SAX. http://www.saxproject.org/about.html, 2004. Último acceso: 27-04-2015.

[57] JSRs: Java Specification Requests. Jsr 222: Java architecture for xml binding (jaxb) 2.0. https://jcp.org/en/jsr/detail?id=222, 2009. Último acceso: 27-04-2015.

[58] E. Rescorla. Http over tls. RFC 2818, RFC Editor, May 2000. `http://www.rfc-editor.org/rfc/rfc2818.txt`.

[59] Inc. RTD Embedded Technologies. Idan description. http://www.rtd.com/IDAN/idan_desc.htm, 2014. Último acceso: 30/05/2015.

[60] I. Rubin and R. Zhang. Placement of uavs as communication relays aiding mobile ad hoc wireless networks. In *Military Communications Conference*, pages 1–7. IEEE, 2007.

[61] RXTX. Página web rxtx project. http://rxtx.qbang.org/wiki/index.php/Rxtx:About, 2009. Último acceso: 30/05/2015.

[62] Christian Schenk. Página web miktex. http://miktex.org/, 2015. Último acceso: 30/05/2015.

[63] Y. Shafranovich. The application/sql media type. RFC 6922, RFC Editor, April 2013. `http://www.rfc-editor.org/rfc/rfc6922.txt`.

[64] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). RFC 7252, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7252.txt`.

[65] Smartsheet. Página web smartsheet. http://es.smartsheet.com/, 2015. Último acceso: 30/05/2015.

[66] Mónica Tilves. La popularidad de php sube como la espuma, aunque c sigue siendo el lenguaje de programación más utilizado. http://www.siliconweek.es/projects/la-popularidad-de-php-sube-como-la-espuma-aunque-c-sigue-siendo-el-lenguaje-de-programacion-mas-utilizado-42806, 2013. Último acceso: 15-04-2015.

[67] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. *SIGMETRICS Perform. Eval. Rev.*, 33(1):291–302, June 2005.

[68] Robert Valdés. How de predator uav works. http://science.howstuffworks.com/predator.htm, 2004 (Último acceso: 07-04-2015).

[69] W3C. Document Object Model (DOM). http://www.w3.org/DOM/, 2005. Último acceso: 27-04-2015.

[70] Oracle webpage. Java EE at a Glance. `http://www.oracle.com/technetwork/java/javaee/overview/index.html`, 2015. Último acceso: 15-04-2015.

[71] T. Ylonen and C. Lonvick. The secure shell (ssh) protocol architecture. RFC 4251, RFC Editor, January 2006. `http://www.rfc-editor.org/rfc/rfc4251.txt`.

[72] Inc. Zend Technologies. User Guide Zend Optimizer v3.3. `http://www.zend.com/en/products/guard/downloads#Windows`, 2007. Último acceso: 15-04-2015.